# **AFL-automation**

**NIST AFL Team** 

## **GETTING STARTED**

1	Installing AFL-automation	3
2	Quick Start Guide	5
3	Tutorials	7
4	How-to Guides	9
5	Explanation	13
6	Reference	15
7	Module Documentation	819
8	Indices and tables	827
Рy	thon Module Index	829
In	dex	831

AFL-automation is a framework for instrument control and laboratory automation. It powers the NIST AFL (Autonomous Formulation Laboratory), but is designed to be versatile for many scientific instrumentation needs. It enables the easy conversion of Python classes - drivers - into robust HTTP microservices with authentication, task queueing, UI generation, data management, and more.

GETTING STARTED 1

2 GETTING STARTED

**CHAPTER** 

ONE

## **INSTALLING AFL-AUTOMATION**

This tutorial will guide you through the process of installing AFL-automation and its dependencies.

### 1.1 Basic Installation

AFL-automation can be installed using pip:

```
pip install AFL-automation
```

This will install the core dependencies needed for basic functionality.

## 1.2 Installation with Hardware Support

Depending on your specific hardware needs, you may want to install additional dependencies:

```
# For Ocean Insight spectrometers
pip install AFL-automation[seabreeze]

# For Opentrons liquid handling robots
pip install AFL-automation[opentrons]

# For multiple hardware types
pip install AFL-automation[seabreeze,opentrons]
```

For a complete list of available extras and what they provide, see the *Managing Dependencies* page.

## 1.3 Development Installation

For development, you might want to install in editable mode with additional tools:

```
git clone https://github.com/usnistgov/AFL-automation.git
cd AFL-automation
pip install -e .

# Install development tools
pip install -e ".[docs]"
```

**CHAPTER** 

**TWO** 

## **QUICK START GUIDE**

This quick start guide will help you get up and running with AFL-automation quickly.

## 2.1 Creating Your First Driver

AFL-automation is built around the concept of *drivers* - Python classes that interact with hardware or provide services. Here's a simple example:

## 2.2 Running as a Microservice

Once you have a driver, you can turn it into a web service:

```
from AFL.automation.APIServer.APIServer import APIServer

# Create the driver instance
my_driver = SimpleDriver()

# Create the server
server = APIServer('my-service')

# Add your driver and run the server
server.create_queue(my_driver)
server.run(host='0.0.0.0', port=5000)
```

## 2.3 Accessing the Service

You can now access your service via HTTP requests or use the built-in client:

```
from AFL.automation.APIServer.Client import Client

# Connect to the service
client = Client('http://localhost:5000')

# Call a method
response = client.enqueue(task_name='say_hello',interactive=True)
print(response) # Outputs: 'Hello, World!'

# Call a method asynchronously
response = client.enqueue(task_name='say_hello',interactive=False)
print(response) # Outputs a uuid

# Get the result of the task
result = client.get_result(response)
print(result) # Outputs: 'Hello, World!'
```

## 2.4 Next Steps

For more detailed instructions, check out:

- AFL-automation Architecture Learn about AFL-automation's architecture
- Managing Dependencies Manage hardware-specific dependencies
- API Reference Explore the full API documentation

## **THREE**

## **TUTORIALS**

Tutorials are learning-oriented guides that help new users get started with AFL-automation.

**CHAPTER** 

**FOUR** 

### **HOW-TO GUIDES**

How-to guides are problem-oriented instructions that help users accomplish specific tasks with AFL-automation.

## 4.1 Managing Dependencies

The AFL-automation package uses a modular dependency system to handle various hardware and specialized libraries. This allows you to install only the dependencies you need for your specific hardware configuration.

## 4.1.1 Core Dependencies

The core dependencies are installed automatically when you install the package. These include:

- Web framework components: Flask, Flask-CORS, Flask-JWT-Extended
- Data processing libraries: NumPy, pandas, SciPy, xarray, h5py
- Visualization tools: Matplotlib, Plotly, Bokeh, ipywidgets
- Scientific utilities: periodictable, scikit-learn, scikit-image

## 4.1.2 Optional Dependencies (Extras)

The package uses Python's "extras" mechanism to manage optional dependencies. You can install these using pip:

```
# Install with specific hardware support
pip install AFL-automation[seabreeze,opentrons]

# Install with scattering support
pip install AFL-automation[scattering-processing,sas-analysis]
```

#### 4.1.3 Available Extras

## **Hardware Interfaces**

Extra Name	Package(s)	Description
labjack	labjack-ljm	Support for LabJack data acquisition hardware
piplates	piplates	Support for Pi-Plates hardware add-ons
rpi-gpio	RPi.GPIO	Raspberry Pi GPIO interface libraries
serial	pyserial	Serial communication support for instruments
seabreeze	seabreeze	Support for Ocean Optics/Ocean Insight spectrometers
opentrons	opentrons	Support for Opentrons liquid handling robots
pyspec	certif-pyspec	Support for CHESS and other beamline control
remote-access	paramiko	SSH client for remote instrument connections

## **Scientific Processing**

Extra Name	Package(s)	Description
scattering-proces	fabio, pyFAI	Scattering data processing and azimuthal integration
sas-analysis	sasmodels, sasdata	Small-Angle Scattering analysis tools
ml	tensorflow, gpflow	Machine learning utilities for data analysis
geometry	alphashape, shapely	Geometric analysis and manipulation tools

## **Neutron Scattering**

Extra Name	Package(s)	Description
neutron-scatterin	epics, sans, mantid	Support for neutron scattering instrumentation
nice-neutron-scat	t nice	NIST NCNR NICE control system

## **Documentation**

Extra Name	Package(s)	Description
docs	sphinx, sphinx-rtd-theme, sphinx-autodoc-typehints, sphinx-copybutton, myst-parser, nbsphinx	Tools for building documentation

### 4.1.4 Implementation Details

AFL-automation uses lazy loading for optional dependencies. This means that:

- 1. You can import the package without having all optional dependencies installed
- 2. Dependencies are only loaded when actually used (via the lazy\_loader library)
- 3. Clear error messages will tell you which extra to install if you try to use a feature without its required dependency

Example of error when trying to use a feature without the required dependency:

```
>>> from AFL.automation.instrument import SeabreezeUVVis
>>> spectrometer = SeabreezeUVVis()
ImportError: This module requires the 'seabreeze' package.
Please install it: pip install AFL-automation[seabreeze]
```

### 4.1.5 Importing Multiple Hardware Packages

If you need support for multiple hardware types, you can specify multiple extras:

```
pip install AFL-automation[seabreeze,opentrons,scattering-processing]
```

Core packages that are not tied to specific hardware will be loaded normally without any special handling.

### **EXPLANATION**

Explanation guides are understanding-oriented articles that provide background context and explain key concepts of AFL-automation.

## 5.1 AFL-automation Architecture

This page explains the core architecture of AFL-automation and how its components work together.

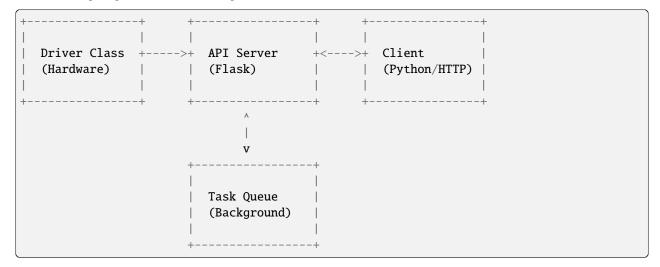
## 5.1.1 Core Concepts

AFL-automation is built around several key concepts:

- 1. Drivers: Python classes that interface with hardware or provide services
- 2. APIServer: A Flask-based web server that exposes drivers via HTTP
- 3. Task Queue: A system for managing asynchronous tasks
- 4. **Client**: A Python client for interacting with remote services

## 5.1.2 Architecture Diagram

The following diagram illustrates the high-level architecture:



## 5.1.3 Driver System

The driver system is the core of AFL-automation. Drivers:

- Encapsulate hardware control logic
- Define configuration parameters with defaults
- Provide methods for interacting with hardware
- · Support lazy loading of hardware-specific dependencies

### 5.1.4 API Server

The API server:

- Exposes driver methods via HTTP endpoints
- · Provides authentication and authorization
- Manages a task queue for asynchronous operations
- · Offers a web UI for monitoring and control

## **5.1.5 Dependency Management**

AFL-automation uses a modular dependency system:

- · Core dependencies are always installed
- Hardware-specific dependencies are optional
- Lazy loading ensures code works even without all dependencies
- The extras system makes installation straightforward

For details on managing dependencies, see Managing Dependencies.

**CHAPTER** 

SIX

## **REFERENCE**

Reference documentation provides information-oriented technical descriptions of the AFL-automation API and code structure.

## 6.1 API Reference

This page provides detailed API documentation for all modules in the AFL-automation framework.

AFL.automation

AFL.automation.APIServer

AFL.automation.instrument

AFL.automation.loading

AFL.automation.prepare

AFL.automation.sample

AFL.automation.sample

## 6.1.1 AFL.automation

#### **Functions**

test() Run all tests using pytest.

AFL.automation.test()

Run all tests using pytest.

## Modules

APIServer

EpicsADLiveProcess

instrument

loading

prepare

sample\_env

shared

## AFL.automation.APIServer

### Modules

APIServer	
Client	
Driver	
DummyDriver	
DummyOT2Driver	
LoggerFilter	
QueueDaemon	

## AFL.automation.APIServer.APIServer

## **Functions**

<pre>create_access_token(identity[, fresh,])</pre>	Create a new access token.
create_refresh_token(identity[,])	Create a new refresh token.
<pre>get_jwt_identity()</pre>	In a protected endpoint, this will return the identity of the JWT that is accessing the endpoint.
<pre>jsonify(*args, **kwargs)</pre>	Serialize the given arguments as JSON, and return a Response object with the application/json mimetype.
<pre>jwt_required([optional, fresh, refresh,])</pre>	A decorator to protect a Flask endpoint with JSON Web Tokens.
listify(obj)	
<pre>render_template(template_name_or_list, **context)</pre>	Render a template by name with the given context.
<pre>send_file(path_or_file[, mimetype,])</pre>	Send the contents of a file to the client.
<pre>set_access_cookies(response,[, max_age,])</pre>	Modifiy a Flask Response to set a cookie containing the access JWT.
<pre>set_refresh_cookies(response,[,])</pre>	Modifiy a Flask Response to set a cookie containing the refresh JWT.
strtobool(val)	Convert a string representation of truth to true (1) or false (0).
unset_jwt_cookies(response[, domain])	Modifiy a Flask Response to delete the cookies containing access or refresh JWTs.

## Classes

APIServer(name[, data, experiment, contact,])	
CORS([app])	Initializes Cross Origin Resource sharing for the application.
FileHandler(filename[, mode, encoding,])	A handler class which writes formatted logging records to disk files.
<pre>Flask(import_name[, static_url_path,])</pre>	The flask object implements a WSGI application and acts as the central object.
IPVersion(value[, names, module, qualname,])	
<pre>JWTManager([app, add_context_processor])</pre>	An object used to hold JWT settings and callback functions for the Flask-JWT-Extended extension.
LoggerFilter(*filters)	
MutableQueue()	Thread-safe, mutable queue
QueueDaemon(app, driver, task_queue, history)	
SMTPHandler(mailhost, fromaddr, toaddrs, subject)	A handler class which sends an SMTP email for each logging event.
ServiceInfo	Service information.
Zeroconf([interfaces, unicast, ip_version,])	Implementation of Zeroconf Multicast DNS Service Discovery

```
class AFL.automation.APIServer.APIServer(name, data=None, experiment='Development',
                                                           contact='tbm@nist.gov',
                                                           index template='index.html',
                                                           new_index_template='index-new.html',
                                                          plot_template='simple-bokeh.html')
     __init__(name, data=None, experiment='Development', contact='tbm@nist.gov',
               index_template='index.html', new_index_template='index-new.html',
               plot_template='simple-bokeh.html')
     create_queue(driver, add_unqueued=True)
     reset_queue_daemon(driver=None)
     advertise_zeroconf(**kwargs)
     run(**kwargs)
     run_threaded(start_thread=True, **kwargs)
     add_standard_routes()
     get_info()
          Live, status page of the robot
     get_quickbar()
          Return the functions, params, and defaults to be shown in this server's quickbar
     is_server_live()
     get_unqueued_commands()
     get_queued_commands()
     add_unqueued_routes()
     query_driver()
     init_logging(toaddrs=None)
     index()
          Live, status page of the robot
     index_new()
          Live, status page of the robot
     webapp()
          Live, status page of the robot
     render_unqueued(func, kwargs_add, **kwargs)
          Convert an unqueued return item into web-suitable output
     send_1d_plot(result, multi=False, **kwargs)
     send_array_as_jpg(array, log_image=False, max_val=None, fillna=0.0, **kwargs)
     queue_state()
     driver_status()
```

```
get_queue()
get_queue_iteration()
deposit_obj()
    Store an object named obj in the driver's dropbox If a unid is provided, the object will be stored with that unid Otherwise, a new unid will be generated. In either case, the unid will be returned to the client.
retrieve_obj()
    Retrieve an object from the driver's dropbox unid specifies the object to retrieve delete specifies whether
```

Retrieve an object from the driver's dropbox unid specifies the object to retrieve delete specifies whether to delete the object after retrieval

```
set_driver_object()
get_driver_object()
enqueue()
reorder_queue()
remove_items()
remove_item()
move_item()
clear_queue()
clear_history()
debug()
pause()
halt()
init()
login()
get_server_time()
login_test()
```

#### AFL.automation.APIServer.Client

#### **Classes**

Client([ip, port, username, interactive])	Communicate with APIServer
ServerDiscovery()	ServerDiscovery class

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
{\tt get\_unqueued\_commands}(inherit\_commands{=}True)
get_queued_commands(inherit commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
```

```
set_driver_object(**kw)
get_driver_object(name)

deposit_obj(obj, uid=None)
    Deposit an object in the dropbox obj : object, the object to deposit id : str, the uuid to deposit the object under if not specified, a new uuid will be generated

retrieve_obj(uid, delete=True)
    Retrieve an object from the dropbox id : str, the uuid of the object to retrieve delete : bool, if True, delete the object after retrieving

set_object(serialize=True, **kw)
get_object(name, serialize=True)
```

#### AFL.automation.APIServer.Driver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
<pre>makeRegistrar()</pre>	
sqrt(x, l)	Return the square root of x.

#### **Classes**

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

• **uid** (*str*) – The uuid to store the object under

#### AFL.automation.APIServer.DummyDriver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x,/)	Return the square root of x.

#### **Classes**

```
Driver(name[, defaults, overrides])
 DummyDriver([name, overrides])
class AFL.automation.APIServer.DummyDriver.DummyDriver(name=None, overrides=None)
     defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
     __init__(name=None, overrides=None)
     status()
     test_command1(kwarg1=None, kwarg2=True)
          A test command with positional and keyword parameters
     test_command2(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     test_command_sets_data(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     how_many(**kwargs)
     test_plot(**kwargs)
     test_image(**kwargs)
     quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
     quickbar_test2()
     dummy_reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     loadSample(cellname='cell', sampleVolume=0)
```

#### AFL.automation.APIServer.DummyOT2Driver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x,/)	Return the square root of x.

#### **Classes**

```
Driver(name[, defaults, overrides])
 DummyDriver([name, overrides])
class AFL.automation.APIServer.DummyOT2Driver.DummyDriver(name=None, overrides=None)
     defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
     __init__(name=None, overrides=None)
     status()
     execute(**kwargs)
     get_prep_target(**kwargs)
     test_command1(kwarg1=None, kwarg2=True)
          A test command with positional and keyword parameters
     test_command2(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     test_command_sets_data(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     how_many(**kwargs)
     test_plot(**kwargs)
     test_image(**kwargs)
     quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
     quickbar_test2()
     dummy_reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     loadSample(cellname='cell', sampleVolume=0)
```

#### AFL.automation.APIServer.LoggerFilter

#### Classes

```
LoggerFilter(*filters)
```

```
class AFL.automation.APIServer.LoggerFilter.LoggerFilter(*filters)
    __init__(*filters)
```

#### AFL.automation.APIServer.QueueDaemon

#### **Functions**

```
is\_serialized (obj)
```

#### Classes

DataTrashcan()	A DataPacket implementation <i>for testing only</i> that takes all its data and simply throws it away.
QueueDaemon(app, driver, task_queue, history)	

```
__init__(app, driver, task_queue, history, debug=False, data=None)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

terminate()
check\_if\_paused()
mask\_serialized\_objs(package)

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### AFL.automation.EpicsADLiveProcess

#### **Modules**

AreaDetectorLive

Client

ReduceDaemon

### AFL.automation.EpicsADLiveProcess.AreaDetectorLive

#### **Classes**

```
AreaDetectorLive([basepv, cam, filewriter, ...])
```

#### AFL.automation.EpicsADLiveProcess.Client

#### **Classes**

```
Client([ip, port]) Communicate with NistoRoboto server on OT-2
```

```
class AFL.automation.EpicsADLiveProcess.Client.Client(ip='10.42.0.30', port='5000')
Communicate with NistoRoboto server on OT-2
```

This class maps pipettor functions to HTTP REST requests that are sent to the NistoRoboto server

```
__init__(ip='10.42.0.30', port='5000')

logged_in()

login(username)

set_queue_mode(debug_mode=True)

transfer(source, dest, volume, source_loc=None, dest_loc=None)

Transfer fluid from one location to another

Parameters

• source(str or list of str) - Source wells to
```

- **source** (*str or list of str*) Source wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **dest** (*str or list of str*) Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **volume** (*float*) volume of fluid to transfer in microliters

```
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
reset()
home()
halt()
```

#### AFL.automation.EpicsADLiveProcess.ReduceDaemon

#### **Classes**

```
ReduceDaemon(app, reduction_queue, ...[, ...])
```

```
__init__(app, reduction_queue, integrator, results, mask=None, npts=500)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### terminate()

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### **AFL**.automation.instrument

#### **Modules**

DummySAS

FileCamera

I22SAXS

NetworkCamera

SeabreezeUVVis

SpecScreen\_Driver

#### AFL.automation.instrument.DummySAS

#### **Classes**

```
Driver(name[, defaults, overrides])

DummySAS([overrides])

class AFL.automation.instrument.DummySAS.DummySAS(overrides=None)

    defaults = {}
    __init__(overrides=None)
        connect to spec

expose(name=None, exposure=None, nexp=1, block=True, reduce_data=True, measure_transmission=True, save_nexus=True)

status()
```

#### AFL.automation.instrument.FileCamera

#### **Classes**

```
FileCamera()

class AFL.automation.instrument.FileCamera.FileCamera
    __init__()
    collect(fname)
```

#### AFL.automation.instrument.I22SAXS

#### Classes

```
Driver(name[, defaults, overrides])

I22SAXS([overrides])

class AFL.automation.instrument.I22SAXS.I22SAXS(overrides=None, **kwargs)

defaults = {'acq_time': 1, 'acq_timeout': 120, 'address': '',
    'data_read_cooldown': 5, 'empty_scan_id': '', 'file_read_timeout': 30,
    'nframes': 1, 'port': 2222, 'pos_list': [], 'processed_base_path':
    '/mnt/i22_processed/i22-', 'reduced_data_suffix':
    '_saxs_Transmission_Averaged_Subtracted_IvsQ_processed.nxs', 'ssh_key_path': '',
    'username': ''}

__init__(overrides=None, **kwargs)

expose(name, empty=False, nframes=None, acq_time=None, set_empty=False)
    Perform a sequence of exposures at positions defined in self.config['pos_list'].
    Return the integrated data of the measurement with the lowest measured sample transmission.
    read_integrated(scanid)
    Scans the appropriate directory for the filename of the data collected with filename
```

#### AFL.automation.instrument.NetworkCamera

#### **Classes**

```
NetworkCamera(url)
```

```
class AFL.automation.instrument.NetworkCamera.NetworkCamera(url)
    __init__(url)
    camera_reset()
    collect()
```

#### AFL.automation.instrument.SeabreezeUVVis

#### **Classes**

```
      Driver(name[, defaults, overrides])

      Eq(key, value)
      Query equality of a given key's value to the specified value.

      Path(*args, **kwargs)
      PurePath subclass that can make system calls.

      SeabreezeUVVis([backend, device_serial, ...])
```

```
defaults = {'air_uuid': '', 'correctDarkCounts': False, 'correctNonlinearity':
False, 'exposure': 0.01, 'exposure_delay': 0, 'filename': 'test.h5', 'filepath':
'.', 'reference_uuid': '', 'saveSingleScan': False}
__init__(backend='cseabreeze', device_serial=None, overrides=None)
getExposure()
getExposureDelay()
getFilename()
getSaveSingleScan()
getFilepath()
setFilepath(filepath)
setFilename(filename)
setSaveSingleScan(saveSingleScan)
setExposureDelay(time)
setExposure(time)
collectContinuous(duration, start=None, return_data=False, **kwargs)
collectSingleSpectrum(set_reference=False, set_air=False, **kwargs)
```

**collect**(nframes: int, reduced: bool = False, absorbance: bool = True, set\_reference: bool = False, set\_air: bool = False, exposure: float | None = None, return\_data: bool = False, \*\*kwargs)

reduced(data\_raw\_mean, data\_raw\_std, absorbance=True, reference\_uuid='reference\_uuid')

#### AFL.automation.instrument.SpecScreen\_Driver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x,/)	Return the square root of x.

#### Classes

```
Driver(name[, defaults, overrides])
SpecScreen_Driver([log_file])
```

```
class AFL.automation.instrument.SpecScreen_Driver.SpecScreen_Driver(log_file=None)
    __init__(log_file=None)
    status()
    execute(**kwargs)
```

AFL.automation.loading

# Modules

CetoniMultiPosValve	
ChemyxSyringePump	This is largely duplicated from the reference code provided by Chemyx.
DigitalOutPressureController	race of Chemyn.
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
LoadStopperDriver	
MultiChannelRelay	
NE1kSyringePump	
OneSelectorBlowoutSampleCell	
PneumaticPressureSampleCell	
PneumaticSampleCell	
PressureController	
PressureControllerAsPump	
PushPullSelectorSampleCell	
RSoXSSolutionSampleCell	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
SyringePump	
Tubing	
TwoSelectorBlowoutSampleCell	
UltimusVPressureController	
- ViciMultiposSelector 6.1. API Reference	33

## AFL.automation.loading.CetoniMultiPosValve

#### **Classes**

```
CetoniMultiPosValve(parentpump[, portlabels])

FlowSelector()
```

```
__init__(parentpump, portlabels={})
connect to valve and query the number of positions
```

#### **Parameters**

- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

#### selectPort(port, direction=None)

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

```
getPort(as_str=False)
    query the current selected position
```

#### AFL.automation.loading.ChemyxSyringePump

This is largely duplicated from the reference code provided by Chemyx. Their package is a GUI and direct import of the module would be problematic.

#### **Functions**

<pre>getOpenPorts()</pre>	
parsePortName(portinfo)	On macOS and Linux, selects only usbserial options and parses the 8 character serial number.

#### **Classes**

```
ChemyxConnection(port, baudrate[, x, mode, ...])
 ChemyxSyringePump(port, syringe_id_mm, ...)
 SyringePump()
class AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump(port, syringe_id_mm,
                                                                             syringe_volume, baud=9600,
                                                                             flow_delay=5)
     __init__(port, syringe_id_mm, syringe_volume, baud=9600, flow_delay=5)
          Initializes and verifies connection to a Chemyx syringe pump.
              port = serial port reference
              syringe_id_mm = syringe inner diameter in mm, used for absolute volume.
                  (will re-program the pump with this diameter on connection)
              syringe_volume = syringe volume in mL
              baud = baudrate for connection
     wait_dosage_finished(timeout_seconds=60)
          The function waits until the last dosage command has finished until the timeout occurs.
     stop()
          Abort the current dispense/withdraw action.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     getLevel()
     setLevel(level)
     blockUntilStatusStopped(pollingdelay=0.2)
          This is a deprecated function from old serial logic. It should work, but do not use.
     getStatus()
          query the pump status and return whether the pump is moving or not (true if moving, false if stopped)
     getValueFromParams(search_key)
AFL.automation.loading.ChemyxSyringePump.getOpenPorts()
AFL.automation.loading.ChemyxSyringePump.parsePortName(portinfo)
```

6.1. API Reference 35

On macOS and Linux, selects only usbserial options and parses the 8 character serial number.

```
__init__(port, baudrate, x=0, mode=0, verbose=False)
openConnection()
closeConnection()
sendCommand(command)
getResponse()
startPump()
stopPump()
pausePump()
restartPump()
setUnits(units)
setDiameter(diameter)
setRate(rate)
setVolume(volume)
setDelay(delay)
setTime(timer)
getParameterLimits()
getParameters()
getDisplacedVolume()
getElapsedTime()
getPumpStatus()
addMode(command)
addX(command)
```

## AFL.automation.loading.DigitalOutPressureController

DigitalOutPressureController(digital_out,)	
PressureController()	Abstract superclass for pressure controllers that provides timed dispensing

```
AFL-automation
class AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController(digital_out,
                                                                                                   pres-
                                                                                                   sure_to_v_conv)
     __init__(digital_out, pressure_to_v_conv)
          Initializes a DigitalOutPressureController
          Params:
              digital_out (AFL.automation.DigitalOut): pressure_to_v_conv (float): pressure units per volt
     set_P(pressure)
AFL.automation.loading.DoubleViciMultiposSelector
Classes
 DoubleViciMultiposSelector(port1, port2[, ...])
 FlowSelector()
 ViciMultiposSelector(port[, baudrate, ...])
class AFL.automation.loading.DoubleViciMultiposSelector.DoubleViciMultiposSelector(port1,
                                                                                              port2,
                                                                                               bau-
                                                                                               drate=9600,
                                                                                              portla-
                                                                                               bels=None)
     __init__(port1, port2, baudrate=9600, portlabels=None)
          connect to valve and query the number of positions
              Parameters
                  • to
                                 (port - string describing the serial port the actuator is
```

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

 ${\tt selectPort}(port, \textit{direction=None})$ 

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

getPort(as\_str=False)

query the current selected position

## AFL.automation.loading.DummyPump

#### **Classes**

```
DummyPump()
 SerialDevice(port[, baudrate, timeout, ...])
 SyringePump()
class AFL.automation.loading.DummyPump.DummyPump
     __init__()
          Dummy pump for testing - does nothing, but boy does it look good doing it.
     stop()
          Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     blockUntilStatusStopped(pollingdelay=0.2)
     getStatus()
          query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
```

## AFL.automation.loading.FlowSelector

```
rlowSelector()

class AFL.automation.loading.FlowSelector.FlowSelector
    getPort()
    selectPort()
```

## AFL.automation.loading.LoadStopperDriver

reset\_stopper()

#### Classes

```
Client([ip, port, username, interactive])

Driver(name[, defaults, overrides])

LoadStopperDriver(sensor[, load_client, ...])

SensorPollingThread(sensor[, period, ...])

StopLoadCBv1(poll, period, load_client[, ...])

StopLoadCBv2(poll, period[, load_client, ...])
```

```
class AFL.automation.loading.LoadStopperDriver.LoadStopperDriver(sensor, load_client=None,
                                                                     load_object=None,
                                                                     auto_initialize=True,
                                                                     overrides=None, data=None,
                                                                     sensorlabel=",
                                                                     name='LoadStopperDriver')
     Driver for stopping loads
     defaults = {'load_speed': 2, 'period': 0.05, 'poll_window': 1000, 'sensorlabel':
     '', 'stopper_baseline_duration': 2, 'stopper_filepath':
     '/github/home/.afl/loadstopper_data', 'stopper_loadstop_cooldown': 2,
     'stopper_min_load_time': 3, 'stopper_post_detection_sleep': 1,
     'stopper_threshold_npts': 50, 'stopper_threshold_std':
     'stopper_threshold_v_step': 1, 'stopper_timeout': 120}
     __init__(sensor, load_client=None, load_object=None, auto_initialize=True, overrides=None, data=None,
              sensorlabel=", name='LoadStopperDriver')
     status()
     property app
     calibrate_sensor()
     read_sensor()
     read_poll()
     read_poll_load()
     reset()
     reset_poll()
```

## AFL.automation.loading.MultiChannelRelay

#### **Classes**

```
MultiChannelRelay()

class AFL.automation.loading.MultiChannelRelay.MultiChannelRelay
   setChannels(channels)
   getChannels(channels)
   toggleChannels(channels)
```

## AFL.automation.loading.NE1kSyringePump

#### **Classes**

```
NE1kSyringePump(port, syringe_id_mm, ...[, ...])
SerialDevice(port[, baudrate, timeout, ...])
SyringePump()
```

```
\begin{tabular}{ll} \textbf{class} & \textbf{AFL}. \textbf{automation.loading.NE1kSyringePump.NE1kSyringePump} (port, syringe\_id\_mm, \\ & syringe\_volume, baud=9600, \\ & daisy\_chain=None, pumpid=None, \\ & flow\_delay=5) \end{tabular}
```

Initializes and verifies connection to a New Era 1000 syringe pump.

port = serial port reference

#### syringe\_id\_mm = syringe inner diameter in mm, used for absolute volume.

(will re-program the pump with this diameter on connection)

syringe\_volume = syringe volume in mL

baud = baudrate for connection

daisy\_chain = used for the 'party-line' mode on these pumps where a string of pumps is on one serial port.

#### when setting up daisy chaining:

connect to the first pump on a port with daisy\_chain = False on subsequent pumps, set daisy\_chain to the pump with a hardware connection (the first pump)

or any other pump on the string.

# note: when daisy chaining you should probably set pumpid explicitly rather than autodiscovering

as most likely the autodiscovery will return the first pump id each time.

**pumpid = the ID configured in the pump firmware. If not set, will attempt to auto-discover a pump.** setting pumpid will save some time on connection and probably result in more reproducible behavior. Practically mandatory for daisy chain mode.

#### stop()

Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.

```
withdraw(volume, block=True, delay=True)
dispense(volume, block=True, delay=True)
setRate(rate)
getRate()
emptySyringe()
blockUntilStatusStopped(pollingdelay=0.2)
getStatus()
```

query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)

## AFL.automation.loading.OneSelectorBlowoutSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
OneSelectorBlowoutSampleCell(pump, selector)	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
<pre>TwoSelectorBlowoutSampleCell(pump, selector,)</pre>	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

class AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell(pump,

```
se-
lec-
tor,
rinse_tank_level=950
waste_tank_level=0,
cell_waste_tank_leve
over-
rides=None)
```

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

```
@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)
```

```
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0, overrides=None)
```

ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells

thickness = cell path length, to be incorporated into metadata, array with length = ncells

cell state if not 'clean', array with length = ncells

## pump: a pump object supporting withdraw() and dispense() methods

e.g. pump = NE1KSyringePump(port,syringe\_id\_mm,syringe\_volume)

# selector: a selector object supporting string-based selectPort() method with options 'catch','cell','rinse','waste','air'

e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})

drySyringe(blow=True, waittime=1)

transfer from air to waste, to push out any residual liquid.

if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.

blowOutCellLegacy(cellname='cell')

**blowOutCell**(cellname='cell', waittime=20)

## AFL.automation.loading.PneumaticPressureSampleCell

Driver(name[, defaults, overrides])	
PneumaticPressureSampleCell(pctrl, relayboard)	Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
SampleCell()	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

```
class AFL.automation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell(pctrl,
                                                                                                      re-
                                                                                                      lay-
                                                                                                      board,
                                                                                                      digi-
                                                                                                      talin=None,
                                                                                                      rinse1 tank level=950,
                                                                                                      rinse2 tank level=950,
                                                                                                      waste\_tank\_level=0,
                                                                                                      load_stopper=None,
                                                                                                      robot_interlock_host=N
                                                                                                      over-
                                                                                                      rides=None)
     Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
     Driven by a variable-pressure regulator.
     defaults = {'arm_move_delay': 0.2, 'blowout_pressure':
      'external_load_complete_trigger': False, 'load_mode': 'static', 'load_pressure':
     2, 'load_timeout': 60, 'ramp_load_duration': 20, 'ramp_load_stop_pressure': 7,
      'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
     0.5), ('blow', 5)], 'vent_delay': 0.5}
     __init__(pctrl, relayboard, digitalin=None, rinsel_tank_level=950, rinse2_tank_level=950,
                waste_tank_level=0, load_stopper=None, robot_interlock_host=None, overrides=None)
          pctrl: a pressurecontroller object supporting the set P method() and the optional p ramp() method.
               e.g. pctrl = DigitalOutPressureController(LabJackDigitalOut(...))
          relayboard: a relay board object supporting string-based setChannels() method
               required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
               e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
     reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     property app
     property data
     status()
     loadSample(cellname='cell', sampleVolume=None, load_dest_label='')
          Load a sample into the cell
          Params cellname and sampleVolume are kept for backward compat, but are deprecated and unused.
     advanceSample(load dest label=")
          Move a sample from one measurement cell to the next
          Params:
               load_dest_label (str, default ''): a 'destination label' for this load.
                   labeled sensors will only stop the load if their name is in this destination label. example:
                     sensor 1 labeled 'afterSANS' sensor 2 labeled 'beforeSPEC' sensor 3 labeled '' (default)
                     advanceSample(load_dest_label='afterSANS') -> sensor 1 or sensor 3 can stop it advance-
                     Sample(load dest label='beforeSPEC afterSANS') -> sensor 1, sensor 2, or sensor 3 can
                     stop it advanceSample(load dest label=") -> only sensor 3 can stop it
```

```
stopLoad(**kwargs)
rinseCell(cellname='cell')
rinseAll()
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
read_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset(sensor_n=None)
```

## AFL.automation.loading.PneumaticSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
<pre>PneumaticSampleCell(pump, relayboard[,])</pre>	Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
SampleCell()	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

```
class AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCell(pump, relayboard,
                                                                               digitalin=None,
                                                                               rinse1_tank_level=950,
                                                                               rinse2_tank_level=950,
                                                                               waste\_tank\_level=0,
                                                                               load_stopper=None,
                                                                               overrides=None)
```

Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.

Driven by a syringe pump.

```
defaults = {'air_speed': 30, 'arm_move_delay': 0.2, 'catch_to_cell_vol': 1.15,
'external_load_complete_trigger': False, 'large_dispense_vol': 5, 'load_speed':
2, 'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
0.5), ('blow', 5)], 'vent_delay': 0.5, 'withdraw_vol': 1.5}
```

```
__init__(pump, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950,
          waste_tank_level=0, load_stopper=None, overrides=None)
     pump: a pump object supporting withdraw() and dispense() methods
         e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
     relayboard: a relay board object supporting string-based setChannels() method
         required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
         e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
reset_pump(dispense=False)
reset_tank_levels(rinse1=950, rinse2=950, waste=0)
property app
status()
loadSample(cellname='cell', sampleVolume=0)
stopLoad(**kwargs)
rinseCell(cellname='cell')
rinseAll()
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
read_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset()
```

#### AFL.automation.loading.PressureController

## **Classes**

PressureController()	Abstract superclass for pressure controllers that provides
	timed dispensing

#### class AFL.automation.loading.PressureController.PressureController

Abstract superclass for pressure controllers that provides timed dispensing

```
timed_dispense(dispense_pressure, dispense_time, block=True)
```

Perform a pressure dispense at pressure *dispense\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop().

```
blockUntilStatusStopped(pollingdelay=0.2)
```

block execution until the controller finishes a dispense

#### dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

#### stop()

Abort the current timed dispense action.

## AFL.automation.loading.PressureControllerAsPump

#### Classes

```
PressureControllerAsPump(pressure_controller)

SerialDevice(port[, baudrate, timeout, ...])

SyringePump()
```

```
\textbf{class} \  \, \textbf{AFL}. \textbf{automation.} \textbf{loading.} \textbf{PressureControllerAsPump.} \textbf{\textit{PressureControllerAsPump}}. \textbf{\textit{PressureControllerAsPump}}. \textbf{\textit{pressure\_controller}}, \\ \textbf{\textit{dis-}}
```

pense\_pressure=3.5, implied flow rate=50)

**\_\_init\_\_**(pressure\_controller, dispense\_pressure=3.5, implied\_flow\_rate=50)

Initialize a pressure controller as a syringe pump.

pressure\_controller: controller to connect to dispense\_pressure: pressure at which dispense should run implied\_flow\_rate: flow rate which should be used to convert to dispense times, mL/min

#### stop()

Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.

```
withdraw(volume, block=True, delay=True)
dispense(volume, block=True, delay=True)
setRate(rate)
getRate()
```

```
emptySyringe()
blockUntilStatusStopped(pollingdelay=0.2)
getStatus()
    query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
```

## AFL.automation.loading.PushPullSelectorSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
PushPullSelectorSampleCell(pump, selector[,])	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
SampleCell()	
Tubing(specid, length)	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

 $\textbf{class} \ AFL. automation. loading. PushPullSelector Sample Cell. \textbf{\textit{PushPullSelectorSampleCell}} (\textit{pump}, \textit{pushPullSelectorSampleCell}) and \textit{pushPullSelectorSampleCell} (\textit{pump}, \textit{pushPullSelectorSampleCell}) and \textit{pushPullSelectorSampleCell}) are the properties of the pro$ 

```
selec-
tor,
ncells=1,
thick-
ness=None.
catch_to_sel_vol=None,
cell_to_sel_vol=None,
ringe_to_sel_vol=None,
selec-
tor_internal_vol=None,
cali-
brated_catch_to_syringe_v
brated_syringe_to_cell_vo
rinse\_speed=50.0,
load\_speed=10.0,
rinse_flow_delay=3.0,
load_flow_delay=10.0)
```

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector) @TODO: figure out when/where to pull in air to the syringe to make up for extra\_vol\_to\_empty\_ml

```
__init__(pump, selector, ncells=1, thickness=None, catch_to_sel_vol=None, cell_to_sel_vol=None,
           syringe_to_sel_vol=None, selector_internal_vol=None, calibrated_catch_to_syringe_vol=None,
          calibrated syringe to cell vol=None, rinse speed=50.0, load speed=10.0,
          rinse_flow_delay=3.0, load_flow_delay=10.0)
     ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
     by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
     thickness = cell path length, to be incorporated into metadata, array with length = ncells
     cell state if not 'clean', array with length = ncells
     pump: a pump object supporting withdraw() and dispense() methods
         e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
     selector: a selector object supporting string-based selectPort() method with options
     'catch','cell','rinse','waste','air'
         e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
property app
status()
setParameter(parameter, value)
getParameters()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCell(cellname='cell')
blowOutCellForcedAir(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
```

## AFL.automation.loading.RSoXSSolutionSampleCell

#### **Classes**

waste\_tank\_level=0, cell\_waste\_tank\_level=0, over-

rides=None)

Class for a sample cell consisting of a pump and a one-to-many flow selector for a flowrate-limited measurement cell (e.g., a liquid TEM sample holder for RSoXS).

The pump draws from any of the sample ports and can quickly purge/rinse itself and the tubing between the loader and the cell. When the pump is connected to the cell, its max rate is limited to very slow (5 ul/min for RSoXS).

```
defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0,
          overrides=None)
    ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
    by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
    thickness = cell path length, to be incorporated into metadata, array with length = ncells
    cell state if not 'clean', array with length = ncells
    pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    selector: a selector object supporting string-based selectPort() method with options
    'catch','cell','rinse','waste','air'
        e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
```

```
reset_tank_levels(rinse=950, waste=0, cell_waste=0)
property app
status()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
setWasteLevel(vol)
```

## AFL.automation.loading.SainSmartRelay

## **Module Attributes**

usbrelay

#### **Classes**

is:

```
MultiChannelRelay()
 SainSmartRelay(relaylabels, serial_port[, ...])
 SerialDevice(port[, baudrate, timeout, ...])
class AFL.automation.loading.SainSmartRelay.SainSmartRelay(relaylabels, serial_port, timeout=0.5)
     __init__(relaylabels, serial_port, timeout=0.5)
          Init connection to a SainSmart 16-channel USB relay module.
          Params: relaylabels (dict):
              mapping of port id to load name, e.g. {0:'arm up',1:'arm down'}
          serial port (str):
              port to connect to
     setAllChannelsOff()
     setChannels(channels)
          Write a value (True, False) to the channels specified in channels
          Parameters: channels (dict):
              dict of channels, keys as either str (name) or int (id) to write to, vals as the value to write
     getChannels(asid=False)
          Read the current state of all channels
          Parameters: asid (bool,default false): Dict keys should simply be the id, not the name.
          Returns: (dict) key:value mappings of state.
     toggleChannels(channels)
AFL.automation.loading.SainSmartRelay.usbrelay = [[b':FE0100200000FF\r\n',
b':FE0100000010F1\r\n'], [b':FE0500000000FD\r\n', b':FE050000FF00FE\r\n'],
[b':FE0500010000FC\r\n', b':FE050001FF00FD\r\n'], [b':FE0500020000FB\r\n',
b':FE050002FF00FC\r\n'], [b':FE0500030000FA\r\n', b':FE050003FF00FB\r\n'],
[b':FE0500040000F9\r\n', b':FE050004FF00FA\r\n'], [b':FE0500050000F8\r\n',
b':FE050005FF00F9\r\n'], [b':FE0500060000F7\r\n', b':FE050006FF00F8\r\n'],
[b':FE0500070000F6\r\n', b':FE050007FF00F7\r\n'], [b':FE0500080000F5\r\n',
b':FE050008FF00F6\r\n'], [b':FE0500090000F4\r\n', b':FE050009FF00F5\r\n'],
[b':FE05000A0000F3\r\n', b':FE05000AFF00F4\r\n'], [b':FE05000B0000F2\r\n',
b':FE05000BFF00F3\r\n'], [b':FE05000C0000F1\r\n', b':FE05000CFF00F2\r\n'],
[b':FE05000D0000F0\r\n', b':FE05000DFF00F1\r\n'], [b':FE05000E0000FF\r\n',
b':FE05000EFF00F0\r\n'], [b':FE05000F0000FE\r\n', b':FE05000FF00FF\r\n'],
[b':FE0F00000010020000E1\r\n', b':FE0F0000001002FFFFE3\r\n']]
     "" Sainsmart 16-Channel 9-36v USB Relay Module (CH341 chip)(sku# 101-70-208) 1. Requires CH341 Win-
     dows driver installed (see http://wiki.sainsmart.com/index.php/101-70-208) 2. The hex table provided by Sains-
     mart requires converting to ASCII chars (see 'usbrelay' below) 3. Format of 'usbrelay' two-dimensional array
```

# usbrelay = [row][ch-off, ch-on] so that the 2nd index selects ON/OFF value while the 1st index selects the array row.

#### Example...

```
status = usbrelay[0][1] # row-0 (status) stat_ret = usbrelay[0][0] # row-0 (status return) ch_1_on = usbrelay[1][1] # row-1 (chan-1 off) ch_1_off = usbrelay[1][0] # row-1 (chan-1 off) ch_16_on = usbrelay[16][1] # row-16 (chan-16 on) ch_16_off = usbrelay[16][0] # row-16 (chan-16 off) all_on = usbrelay[17][1] # row-17 (all on) all_off = usbrelay[17][0] # row-17 (all off)
```

٠,,,,

## AFL.automation.loading.SampleCell

#### **Classes**

```
SampleCell()

class AFL.automation.loading.SampleCell.SampleCell
    loadSample()
```

## AFL.automation.loading.Sensor

```
DummySensor1([period, minval, maxval])

DummySensor2([period, hi_value, lo_time, ...])

Sensor([address, channel])

class AFL.automation.loading.Sensor.Sensor(address=1, channel=0)
    __init__(address=1, channel=0)
    calibrate()
    read()

class AFL.automation.loading.Sensor.DummySensor1(period=2, minval=-5, maxval=5)
    __init__(period=2, minval=-5, maxval=5)

    This constructor should always be called with keyword arguments. Arguments are:
        group should be None; reserved for future extension when a ThreadGroup class is implemented.
        target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
        name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.
```

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### read()

#### terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

**class** AFL.automation.loading.Sensor.**DummySensor2**(period=0.1, hi\_value=5, lo\_time=15, hi\_time=2)

```
__init__(period=0.1, hi_value=5, lo_time=15, hi_time=2)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### driver\_status()

#### read()

## terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

# ${\bf AFL.} automation. Io ading. Sensor Callback Thread$

#### **Classes**

```
LoaderCommunication([load_client, load_object])

SensorCallbackThread(poll[, period, daemon, ...])

SimpleThresholdCB(poll, period[, window, ...])

StopLoadCBv1(poll, period, load_client[, ...])

StopLoadCBv2(poll, period[, load_client, ...])
```

```
class AFL.automation.loading.SensorCallbackThread.SensorCallbackThread(poll, period = 0.1, daemon = True, filepath = None, data = None, sensorlabel = ")
```

```
__init__(poll, period=0.1, daemon=True, filepath=None, data=None, sensorlabel=")
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

```
update_status(value)

terminate()

process_signal(signal)

run()
```

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

data=None, sensorlabel=")

```
__init__(poll, period, load_client, threshold_npts=20, threshold_v_step=1, threshold_std=2.5, timeout=120, loadstop_cooldown=2, post_detection_sleep=0.2, baseline_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

# process\_signal()

class AFL.automation.loading.SensorCallbackThread.StopLoadCBv2(poll, period, load\_client=None,

load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### process\_signal()

```
__init__(poll, period, window=5, threshold=1)
           This constructor should always be called with keyword arguments. Arguments are:
           group should be None; reserved for future extension when a ThreadGroup class is implemented.
           target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
           name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a
           small decimal number.
           args is a list or tuple of arguments for the target invocation. Defaults to ().
           kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.
           If a subclass overrides the constructor, it must make sure to invoke the base class constructor
           (Thread.__init__()) before doing anything else to the thread.
     process_signal()
class AFL.automation.loading.SensorCallbackThread.LoaderCommunication(load_client=None,
                                                                                      load_object=None)
     __init__(load_client=None, load_object=None)
     getServerState()
     stopLoad()
AFL.automation.loading.SensorPollingThread
Classes
 SensorPollingThread(sensor[, period, ...])
class AFL.automation.loading.SensorPollingThread.SensorPollingThread(sensor, period=0.1,
                                                                                    callback=None,
                                                                                    hv pipe=None,
                                                                                    window=None,
                                                                                    filename=None,
                                                                                    daemon=True,
                                                                                    data=None)
     __init__(sensor, period=0.1, callback=None, hv_pipe=None, window=None, filename=None,
                daemon=True, data=None)
           This constructor should always be called with keyword arguments. Arguments are:
           group should be None; reserved for future extension when a ThreadGroup class is implemented.
           target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
           name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a
           small decimal number.
           args is a list or tuple of arguments for the target invocation. Defaults to ().
           kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.
```

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

```
read()
read_load_buffer()
reset_load_buffer()
terminate()
alive()
run()
```

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## AFL.automation.loading.SerialDevice

#### **Classes**

```
SerialDevice(port[, baudrate, timeout, ...])
```

## **Exceptions**

SerialCommsException	Raised when the system receives a serial response it can't
	parse, likely a garbled line

```
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
```

## AFL.automation.loading.SyringePump

## **Classes**

```
SyringePump()
```

class AFL.automation.loading.SyringePump.SyringePump

```
stop()
withdraw(volume, block=True)
dispense(volume, block=True)
setRate(rate)
getRate(rate)
emptySyringe()
```

## AFL.automation.loading.Tubing

#### **Classes**

```
Tubing(specid, length)

class AFL.automation.loading.Tubing.Tubing(specid, length)

tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
    'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
    'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in': 0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID_mm': 1, 'OD_in': 0.075, 'material': 'Tefzel', 'typeid': 1517}]
    __init__(specid, length)
    length is in cm?

volume()
    returns volume in mL
```

## AFL.automation.loading.TwoSelectorBlowoutSampleCell

```
      Driver(name[, defaults, overrides])

      SampleCell()

      Tubing(specid, length)

      TwoSelectorBlowoutSampleCell(pump, ...)
      selector, Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

      defaultdict

      defaultdict
      defaultdict(default_factory=None, /, [...]) --> dict with default factory
```

```
class AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell(pump,
                                                                                                     se-
                                                                                                     lec-
                                                                                                     tor,
                                                                                                     blows-
                                                                                                     e-
                                                                                                     lec-
                                                                                                     tor,
                                                                                                     rinse tank level=950
                                                                                                     waste_tank_level=0,
                                                                                                     cell_waste_tank_leve
                                                                                                     over-
                                                                                                     rides=None)
     Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample
     (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell
     where the cell is in the pump line).
     @TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)
     defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
     'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
      'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
     1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
     'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
     'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
     'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
     'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
     1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
     __init__(pump, selector, blowselector, rinse_tank_level=950, waste_tank_level=0,
                cell waste tank level=0, overrides=None)
          ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
          by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
          thickness = cell path length, to be incorporated into metadata, array with length = ncells
          cell state if not 'clean', array with length = ncells
          pump: a pump object supporting withdraw() and dispense() methods
              e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
          selector: a selector object supporting string-based selectPort() method with options
          'catch', 'cell', 'rinse', 'waste', 'air'
              e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
     reset_tank_levels(rinse=950, waste=0, cell_waste=0)
     property app
     status()
     transfer(source, dest, vol source, vol dest=None)
     catchToSyringe(sampleVolume=0)
     loadSample(cellname='cell', sampleVolume=0)
```

6.1. API Reference 59

catchToWaste(sampleVolume=0.0)

## AFL.automation.loading.UltimusVPressureController

#### **Classes**

```
PressureController()
Abstract superclass for pressure controllers that provides timed dispensing
UltimusVPressureController(port[, baud])
```

 $\textbf{class} \hspace{0.1cm} \textbf{AFL.automation.loading.UltimusVPressureController.UltimusVPressureController} (port, \\ baud=115200)$ 

```
__init__(port, baud=115200)
    Initializes a DigitalOutPressureController
    Params:
        port (str): serial port to use

compute_checksum(cmd)

char_count(cmd)

package_cmd(cmd)

send_command(cmd)

set_P(pressure)

pressure: pressure to set in psi
```

## AFL.automation.loading.ViciMultiposSelector

#### **Classes**

```
FlowSelector()

SerialDevice(port[, baudrate, timeout, ...])

ViciMultiposSelector(port[, baudrate, ...])
```

```
__init__(port, baudrate=9600, portlabels=None)
connect to valve and query the number of positions
```

#### **Parameters**

- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

selectPort(port, direction=None, block=True)

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

#### **Parameters**

- port (String or int) the name, or number, of the port to switch to
- **direction** (*String*, *default=None*) direction "CW" or "CCW" to perform the move in
- **block** (*bool*, *default=True*) block return until move completes

```
getPort(as_str=False)
```

query the current selected position

#### AFL.automation.prepare

# **Functions**

```
#D20Factory(name[, phi_D2O, sld, properties]) Create a list of H2O/D2O solutions

compositionSweepFactory(name, components, ...)

make_locs(slot, nrows, ncols)

make_wellplate_locs(slot, size)
```

ComponentDB([path])	
Deck()	
MassBalance()	
<pre>PipetteAction(source, dest, volume[,])</pre>	
<pre>Sample(name, target[, target_check, balancer])</pre>	
SampleSeries()	
Solute(name[, description, density,])	Specialization class for solute components
Solution(name, components[, properties])	
Solvent(name, density[, description,])	Specialization class for solute components

## Modules

Component

DeckBuilderWidget

Dummy\_OT2\_Driver

OT2Client

PrepType

PrepareWidget

SampleSeriesWidget

StockBuilderWidget

SweepBuilderWidget

factory

utilities

# AFL.automation.prepare.Component

# **Functions**

enforce_units(value, unit_type)	Ensure that a number has units and convert to the de-
	fault_units

## Classes

Component(name, description)	Base class for all materials
<pre>PrepType(value[, names, module, qualname,])</pre>	

## **Exceptions**

```
ParseException(pstr[, loc, msg, elem]) Exception thrown when a parse expression doesn't match the input string
```

```
class AFL.automation.prepare.Component.Component(name, description)
     Base class for all materials
     This class defines all of the basic properties and methods to be shared across material objects
     __init__(name, description)
     emit()
     __hash__()
          Needed so Components can be dictionary keys
     copy()
     __iter__()
          Dummy iterator to mimic behavior of Mixture.
     property mass
     set_mass(value)
          Setter for inline mass changes
     property volume
     set_volume(value)
          Setter for inline volume changes
     property density
     property formula
     property moles
     property sld
     property is_solute
     property is_solvent
```

## AFL.automation.prepare.DeckBuilderWidget

## **Functions**

sqrt(x, l) Return the square root of x.

## Classes

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean value in the form of a checkbox.
<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
DeckBuilderWidget()	
<pre>DeckBuilderWidget_Model()</pre>	
<pre>DeckBuilderWidget_View()</pre>	
Dropdown(**kwargs)	Allows you to select a single item from a dropdown.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible box model.

class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget

```
__init__()
     build_deck_object()
     get_deck_config()
     set_deck_config(config)
     send_deck_cb(event)
     save_cb(event)
     load_cb(event)
     update_deck_graphic_cb(event)
     start()
{\bf class} \  \, {\tt AFL.automation.prepare.DeckBuilderWidget.} \\ {\bf DeckBuilderWidget\_Model} \\
     __init__()
     send_deck_config(pi_ip='piot2', align_script='/home/nistoroboto/align.py')
     build_deck_object(config)
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_View
     __init__()
     create_expanded_button(description, button_style)
     draw_deck()
     start()
```

## AFL.automation.prepare.Dummy\_OT2\_Driver

```
Driver(name[, defaults, overrides])
 Dummy_OT2_Driver([overrides])
class AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driver(overrides=None)
     defaults = {'execute_delay': 1}
     __init__(overrides=None)
     reset_prep_targets()
     add_prep_targets(targets, reset=False)
     get_prep_target()
     status()
     reset_tipracks(mount='both')
     home(**kwargs)
     parse_well(loc)
     get_wells(locs)
     get_labware(slot)
     load_labware(name, slot, module=None, **kwargs)
     set_temp(slot, temp)
          Set the temperature of a tempdeck in slot slot
     get_temp(slot)
         Get the temperature of a tempdeck in slot slot
     deactivate_temp(slot)
         Disablea tempdeck in slot slot
     set_shake(rpm)
     stop_shake()
     set_shaker_temp(temp)
     unlatch_shaker()
     latch_shaker()
     get_shaker_temp()
     get_shake_rpm()
```

```
get_shake_latch_status()
load_instrument(name, mount, tip_rack_slots, **kwargs)
transfer(source, dest, volume, *args, **kwargs)
set_aspirate_rate(rate=150)
set_dispense_rate(rate=300)
set_gantry_speed(speed=400)
```

## AFL.automation.prepare.OT2Client

#### **Classes**

<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
<pre>OT2Client([ip, port, username, interactive])</pre>	Communicate with AFL-automation server on OT-2
<pre>PipetteAction(source, dest, volume[,])</pre>	

Communicate with AFL-automation server on OT-2

This class maps pipettor functions to HTTP REST requests that are sent to the AFL-automation server

transfer(source, dest, volume, interactive=None, \*\*kwargs)

Transfer fluid from one location to another

## **Parameters**

- source (str or list of str Source wells to transfer from. Wells should be specified as three) character strings with the first character being the slot number.
- **dest** (*str or list of str*) Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **volume** (*float*) volume of fluid to transfer in microliters

```
reset_tipracks(mount='both')
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
aspirate_rate(rate)
dispense_rate(rate)
home()
```

# AFL.automation.prepare.PrepType

#### **Functions**

```
makeRegistar()

prepRegistrar(prepType)
```

## Classes

Enum(value[, names, module, qualname, type,])	Create a collection of name/value pairs.
<pre>PrepType(value[, names, module, qualname,])</pre>	
auto([value])	Instances are replaced with an appropriate value in Enum class suites.

 $\begin{tabular}{ll} \textbf{class} & \texttt{AFL}. \textbf{automation.prepare.PrepType}. \textbf{PrepType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)} \\ \end{tabular}$ 

BaseComponent = 1

BaseMixture = 2

Solute = 3

Solvent = 4

Solution = 5

# AFL.automation.prepare.PrepareWidget

## **Functions**

Sqrt(x,t) Return the square root of x.	<pre>sqrt(x,/)</pre>	Return the square root of x.
--	----------------------	------------------------------

# Classes

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean <i>value</i> in the form of a checkbox.
DeckBuilderWidget()	
Dropdown(**kwargs)	Allows you to select a single item from a dropdown.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
PrepareWidget()	
PrepareWidget_Model()	
PrepareWidget_View()	
SampleSeriesWidget(deck)	
StockBuilderWidget(deck)	
SweepBuilderWidget(deck)	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible box model.

 $\textbf{class} \ \texttt{AFL}. automation. prepare. Prepare \texttt{Widget}. \textbf{PrepareWidget}$ 

```
__init__()
SweepBuilder_reset_cb(event)
StockBuilder_reset_cb(event)
SampleSeriesTool_reset_cb(event)
start()
class AFL.automation.prepare.PrepareWidget.PrepareWidget_Model
class AFL.automation.prepare.PrepareWidget.PrepareWidget_View
    start(deck_builder_widget)
```

# AFL.automation.prepare.SampleSeriesWidget

## **Functions**

sqrt(x, l)	Return the square root of x.
------------	------------------------------

## **Classes**

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean value in the form of a checkbox.
<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
FloatText(**kwargs)	Displays a float value within a textbox.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible box model.
IntText(**kwargs)	Textbox widget that represents an integer.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
SampleSeriesWidget(deck)	
SampleSeriesWidget_Model(deck)	
SampleSeriesWidget_View()	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible box model.

# $\textbf{class} \ \texttt{AFL}. automation. prepare. Sample Series \texttt{Widget}. \textbf{Sample Series Widget} (\textit{deck})$

```
__init__(deck)
apply_protocol_order_cb(event)
apply_protocol_order()
update_protocol_order_preview_cb(event)
make_protocol()
make_catch_protocol()
submit_cb(event)
build_label(index)
example_label_cb(event)
make_all_labels_cb(event)
make_all_labels()
sync_to_prepare_cb(event)
```

```
reset_uuid_cb(event)
update_sort_order_cb(event, direction)
make_mixing_wells()
update_mixing_well_preview_cb(event)
submit_mixing_wells_cb(event)
start()
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Model(deck)
__init__(deck)
adjust_protocol_order(mix_order)
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_View
__init__()
make_component_grid(components, nsamples)
make_pipette_params(name)
make_mixing_wells()
start(components, nsamples, stock_names)
```

# AFL.automation.prepare.StockBuilderWidget

#### **Functions**

```
sqrt(x, l) Return the square root of x.
```

#### **Classes**

```
StockBuilderWidget_Model(deck)

StockBuilderWidget_View()

class AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget(deck)

__init__(deck)

analyze_stocks_cb(event)

get_stock_objects()
```

```
add_stocks_to_deck()
    get_stock_values()
    save_cb(event, pkl=True)
    load_cb(event)
    update_location_check_cb(event)
    make_stock_cb(event)
    remove_stock_cb(event)
    set_units_cb(event, units)
    start()
class AFL.automation.prepare.StockBuilderWidget_Model(deck)
    __init__(deck)
    add_stocks_to_deck(all_stocks_dict)
    to_stock_objects(all_stocks_dict)
class AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget_View
    __init__()
    make_stock_tab(stock_name, components)
    start()
```

# AFL.automation.prepare.SweepBuilderWidget

## **Functions**

sqrt(x,/)

Return the square root of x.

# Classes

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean value in the form of a checkbox.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible
	box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
SweepBuilderWidget(deck)	
SweepBuilderWidget_Model(deck)	
<pre>SweepBuilderWidget_View()</pre>	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible box model.

 $\textbf{class} \ \texttt{AFL}. automation. prepare. SweepBuilderWidget}. \textbf{SweepBuilderWidget}(\textit{deck})$ \_\_init\_\_(deck) plot\_binary\_cb(click) plot\_ternary\_cb(click) calc\_sweep\_cb(click) validate\_sweep\_cb(click) get\_sweep\_data() get\_deck() update\_component\_row\_cb(event) start()  $\textbf{class} \ \texttt{AFL}. automation. prepare. SweepBuilderWidget. \textbf{SweepBuilderWidget\_Model} (\textit{deck})$ \_\_init\_\_(deck) validate\_sweep(tolerance, progress=None) calc\_sweep(sweep\_dict, progress) class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget\_View \_\_init\_\_() make\_stock\_grid(component\_names) make\_ternary\_plot(component\_names) make\_binary\_plot(component\_names) start(component\_names)

## AFL.automation.prepare.factory

#### **Functions**

## **Classes**

```
Solution(name, components[, properties])
product product(*iterables, repeat=1) --> product object
```

AFL.automation.prepare.factory.**HD20Factory**(name, phi\_D2O=None, sld=None, properties=None)
Create a list of H2O/D2O solutions

AFL.automation.prepare.factory.compositionSweepFactory(name, components,  $vary\_components$ , lo, hi, num, logspace=False, properties=None, progress=None)

# AFL.automation.prepare.utilities

#### **Functions**

```
make_locs(slot, nrows, ncols)
make_wellplate_locs(slot, size)
```

AFL.automation.prepare.utilities.make\_locs(slot, nrows, ncols)

AFL.automation.prepare.utilities.make\_wellplate\_locs(slot, size)

# AFL.automation.sample\_env

#### **Modules**

```
TemperatureDeck
```

# AFL.automation.sample\_env.TemperatureDeck

#### Classes

```
Driver(name[, defaults, overrides])

TemperatureDeck([overrides])

class AFL.automation.sample_env.TemperatureDeck.TemperatureDeck(overrides=None)

defaults = {'serial_port': '/dev/ttyACMO', 'temperature_move_sleep': 0.2,
    'temperature_move_timeout': 900}

__init__(overrides=None)

set_temp(sp)

move_temp(temperature, wait=30, tolerance=0.1)

status()

read_temp()
```

# AFL.automation.shared

#### **Modules**

DataLabelerWidget

DatasetWidget

DiffractionLabeler

MutableQueue

PersistentConfig

ServerDiscovery

exceptions

serialization

units

utilities

widgetui

# AFL.automation.shared.DataLabelerWidget

## **Functions**

sqrt(x, l)	Return the square root of x.
------------	------------------------------

# **Classes**

```
DataLabelerModel(dataset)

DataLabelerView()

DataLabelerWidget(input_dataset, ...[, ...])

OrdinalEncoder(*[, categories, dtype, ...])

Encode categorical features as an integer array.
```

```
\textbf{class} \ \texttt{AFL}. \textbf{automation.} \textbf{shared.} \textbf{DataLabelerWidget.} \textbf{\textit{DataLabelerWidget}} (\textit{input\_dataset: Dataset}, \textit{total automation.}) \textbf{\textit{class}} (\textit{input\_dataset: Dataset, input\_dataset.}) \textbf{\textit{class}} (\textit{input\_dataset.}) \textbf{\textit{class}} (\textit{input\_dataset.})
```

```
sas_variable: str,
composition_variable: str |
List[str], sample_dim: str =
'sample', fit_variable: str | None
= None)
```

\_\_init\_\_(input\_dataset: Dataset, sas\_variable: str, composition\_variable: str | List[str], sample\_dim: str = 'sample', fit\_variable: str | None = None)

#### **Parameters**

- dataset (xr.Dataset) Dataset from AFL
- **sas\_variable** (*str*) Name of data variable in the *xarray.Dataset* that holds the scattering data
- **composition\_variable** (*str | List[str]*) Name of data variable in the *xar-ray.Dataset* that holds the composition. If the composition is split across multiple variables, pass in a list of variables.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to each sample or measurement. This is typically named 'sample' in much of the AFL agent codebase.
- **fit\_variable** (*Optional[str]*) If not none, this data will be plotted along with the sas\_variable data. This data variable should have the same shape as sas\_variable.

```
next_button_callback(click)
     prev_button_callback(click)
     goto_callback(click)
     composition_click_callback(figure, location, click)
     update_plot()
     draw_peaks(peaks)
     change_qstar_callback(figure, location, click)
     change_model_callback(data)
     change_norder_callback(data)
     label(label)
     run()
class AFL.automation.shared.DataLabelerWidget.DataLabelerModel(dataset: Dataset)
     __init__(dataset: Dataset)
     ordinal_phase_labels()
     init_models()
     get_peaks(model, qstar=None, max order=4)
class AFL.automation.shared.DataLabelerWidget.DataLabelerView
     __init__()
     update_plot(x, y, composition)
     remove_vertical_lines()
```

```
add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
update_composition_colors(colors)
run(x, y, all_compositions, composition, models, ternary, components)
```

# AFL.automation.shared.DatasetWidget

#### **Classes**

```
__init__(dataset: Dataset, sample_dim: str = 'sample', scatt_variables: List[str] | None = None, comps_variable: str | None = None, comps_color_variable: str | None = None, xmin: float = 0.001, xmax: float = 1.0)
```

Interactive widget for viewing compositionally varying scattering data

#### **Parameters**

- **dataset** (*xr.Dataset*) *xarray.Dataset* containing scattering data and compositions to be plotted.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to sample variation, typically "sample"
- **comps\_variable** (*Optional[str]*) The name of the *xarray* variable to plot as compositional data. Optional, if not specified, can be customized in the GUI.

Only the first two columns of the data will be used in the plot. If the compositions are in separate `xarray.DataArray`s, they should be grouped into single `xarray.DataArray`s like so:

```
`python ds['comps'] = ds[['A','B','C']].to_array('component').
transpose(...,'component') `
```

- **comps\_color\_variable** (*Optional[str]*) The name of the *xarray* variable to use as the colorscale of the compositional data scatter plot. Optional, if not specified, can be customized in the GUI.
- xmin (float) Set the default q-range of the scattering data. Can be customized in the GUI

```
• xmax (float) – Set the default q-range of the scattering data. Can be customized in the
                   GUI

    Usage

                 • ```python -
                 • DatasetWidget(ds) (widget =)
                 widget.run()
                 • ``` _
     next_button_callback(*args)
     prev_button_callback(*args)
     goto_callback(*args)
     composition_click_callback(figure, location, click)
     update_composition_plot()
     update_scattering_plot()
     update_plots()
     get_comps()
     initialize_plots(*args)
     update_colors(*args)
     apply_sel(*args)
     apply_isel(*args)
     extract_var(*args)
     combine_vars(*args)
     reset_dataset(*args)
     update_dropdowns(*args)
     update_sample_dim(*args)
     update_extract_coords(change)
     run()
class AFL.automation.shared.DatasetWidget.DatasetWidget_Model(dataset: Dataset, sample_dim: str)
     __init__(dataset: Dataset, sample_dim: str)
     property dataset
     reset_dataset()
     split_vars()
         Heuristically try to split vars into categories
```

```
get_non_sample_dims(var: str)
     apply_sel(kw)
     apply_isel(kw)
     combine_vars(combined_var: str, to_combine_vars: List[str])
     extract_var(extract_from_var: str, extract_from_coord: str)
     get_composition(variable)
     get_scattering(variable, index)
class AFL.automation.shared.DatasetWidget.DatasetWidget_View(initial_scatt_variables: List[str] |
                                                                         None = None,
                                                                         initial_comps_variable: str | None =
                                                                         None, initial_comps_color_variable:
                                                                         str \mid None = None)
     __init__(initial_scatt_variables: List[str] | None = None, initial_comps_variable: str | None = None,
                initial_comps_color_variable: str | None = None)
     update_colorscale(colors=None)
     update_selected(**kw)
     {\tt update\_dropdowns}({\it sample\_vars=None}, {\it scatt\_vars=None}, {\it comp\_vars=None})
     plot_sas(x, y, name='SAS', append=False)
     plot\_comp(x, y, z=None, xname='x', yname='y', zname='z', colors=None)
     init_plots()
     init_buttons()
     init_checkboxes()
     init_dropdowns()
     init_inputs()
     run()
```

# AFL.automation.shared.DiffractionLabeler

## **Functions**

sqrt(x, /)

Return the square root of x.

## **Classes**

```
DiffractionLabeler(saxs_data, ...)
 DiffractionLabelerModel(saxs_data, ...)
 DiffractionLabelerView()
 OrdinalEncoder(*[, categories, dtype, ...])
                                                   Encode categorical features as an integer array.
class AFL.automation.shared.DiffractionLabeler.DiffractionLabeler(saxs_data, composition_data,
                                                                          possible_phase_labels)
     __init__(saxs_data, composition_data, possible_phase_labels)
     next_button_callback(click)
     prev_button_callback(click)
     goto_callback(click)
     ternary_click_callback(figure, location, click)
     update_plot()
     draw_peaks(peaks)
     change_qstar_callback(figure, location, click)
     change_model_callback(data)
     change_norder_callback(data)
     label(label)
     run()
class AFL.automation.shared.DiffractionLabeler.DiffractionLabelerModel(saxs_data,
                                                                               composition data,
                                                                               possible_phase_labels)
     __init__(saxs_data, composition_data, possible_phase_labels)
     ordinal_phase_labels()
     init_models()
     get_peaks(model, qstar=None, max_order=4)
class AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView
     __init__()
     update_plot(x, y, composition)
     remove_vertical_lines()
```

```
add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
update_ternary_colors(colors)
run(x, y, all_compositions, composition, models, possible_phase_labels)
```

## AFL.automation.shared.MutableQueue

## **Classes**

MutableQueue()	Thread-safe, mutable queue

# **Exceptions**

Empty	Exception raised by Queue.get(block=0)/get_nowait().
Full	Exception raised by Queue.put(block=0)/put_nowait().

# class AFL.automation.shared.MutableQueue.MutableQueue

Thread-safe, mutable queue

Unlike the standard library CPython queue, this class supportes positional inserts, deletions, and reordering. The tradeoff is performance for both reads and writes to the queue.

## AFL.automation.shared.PersistentConfig

#### Classes

MutableMapping()	A MutableMapping is a generic container for associating key/value pairs.
<pre>PersistentConfig(path[, defaults,])</pre>	A dictionary-like class that serializes changes to disk

class AFL.automation.shared.PersistentConfig.PersistentConfig(path, defaults=None,

overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

A dictionary-like class that serializes changes to disk

This class provides dictionary-like setters and getters (e.g., [] and .update()) but, by default, all modifications are written into a file in json format with the root keys as timestamps. Modifications can be blocked by locking the config, and writing to disk can be disabled by setting the appropriate member attributes (see constructor). On instantiation, if provided with a previously saved configuration file, PersistentConfig will load the file's contents into memory and use the more recent configuration.

```
__init__(path, defaults=None, overrides=None, lock=False, write=True, max_history=10000, datetime_key_format='%y/%d/%m %H:%M:%S.%f')
```

#### Constructor

#### **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

```
__getitem__(key)
Dictionary-like getter via config["param"]
__setitem__(key, value)
Dictionary-like setter via config["param"]=value
```

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

### toJSON()

Serialize the config to json

#### update(update\_dict)

Update several values in config at once

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

#### revert(nth=None, datetime key=None)

Revert config to a historical config

#### **Parameters**

- **nth** (int, **optional\***) Integer index of historical value to revert to. Can be negative to count from end of history. Note that -1 will correspond to the current config.
- datetime\_key (str, optional) datetime formatted string as defined by datetime\_key\_format

```
get_historical_values(key, convert_to_datetime=False)
```

Convenience method for gathering historical values of a parameter

## AFL.automation.shared.ServerDiscovery

#### **Classes**

AsyncServiceBrowser(zeroconf, type_[,])	Used to browse for a service for specific type(s).
AsyncServiceInfo	An async version of ServiceInfo.
AsyncZeroconf([interfaces, unicast,])	Implementation of Zeroconf Multicast DNS Service Discovery
AsyncZeroconfServiceTypes()	An async version of ZeroconfServiceTypes.
<pre>IPVersion(value[, names, module, qualname,])</pre>	
RunThread(coro)	
ServerDiscovery()	ServerDiscovery class
ServiceBrowser(zc, type_[, handlers,])	Used to browse for a service of a specific type.
ServiceInfo	Service information.
ServiceStateChange(value[, names, module,])	
Zeroconf([interfaces, unicast, ip_version,])	Implementation of Zeroconf Multicast DNS Service Discovery

# class AFL.automation.shared.ServerDiscovery.RunThread(coro)

```
__init__(coro)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### class AFL.automation.shared.ServerDiscovery.ServerDiscovery

ServerDiscovery class

This class is used to discover AFL servers on the network using zeroconf requests that match a particular specification.

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

#### async classmethod sa\_aio\_discover\_server\_by\_name(service name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## classmethod sa\_discover\_server\_by\_name(service\_name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## find\_server\_by\_name(service\_name)

Disambiguator for either matching or discovering a specific server by name

# match\_server\_by\_name(service\_name)

Looks through the registry of discovered services for an exact name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

## find\_server\_by\_partial\_name(service\_name)

Looks through the registry of discovered services for a partial name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

# find\_server\_by\_property\_match(property\_name, property\_value)

Looks through the registry of discovered services for a partial match in a property string. Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

# AFL.automation.shared.exceptions

# **Exceptions**

EmptyException	Raised when a mixture runs out of mass or volume
MixingException	Raised when a mixture cannot be made
NoDeviceFoundException	Raised when no matching device can be found on the selected port
NotFoundError	
SerialCommsException	Raised when the system receives a serial response it can't parse, likely a garbled line

# exception AFL.automation.shared.exceptions.EmptyException

Raised when a mixture runs out of mass or volume

exception AFL.automation.shared.exceptions.MixingException

Raised when a mixture cannot be made

# $\textbf{exception} \hspace{0.1cm} \textbf{AFL}. automation. shared. \textbf{exceptions}. \textbf{Serial Comms Exception} \\$

Raised when the system receives a serial response it can't parse, likely a garbled line

# $\textbf{exception} \ \, \textbf{AFL}. automation. shared. \textbf{exceptions}. \textbf{NoDeviceFoundException}$

Raised when no matching device can be found on the selected port

exception AFL.automation.shared.exceptions.NotFoundError

#### AFL.automation.shared.serialization

# **Functions**

deserialize(pickled\_str)

is\_serialized(obj)

serialize(obj)

# **Exceptions**

UnpicklingError

AFL.automation.shared.serialization.serialize(obj)

AFL.automation.shared.serialization.is\_serialized(obj)

AFL.automation.shared.serialization.deserialize(pickled\_str)

# AFL.automation.shared.units

#### **Functions**

```
AFL.automation.shared.units.is_volume(value)
```

AFL.automation.shared.units.has\_units(value)

 ${\tt AFL.automation.shared.units.} \textbf{is\_molarity}(\textit{value})$ 

AFL.automation.shared.units.**is\_mass**(value)

AFL.automation.shared.units.is\_density(value)

AFL.automation.shared.units.is\_concentration(value)

AFL.automation.shared.units.get\_unit\_type(value)

AFL.automation.shared.units.enforce\_units(value, unit\_type)
Ensure that a number has units and convert to the default\_units

# AFL.automation.shared.utilities

# **Functions**

```
has_units(value)
listify(obj)

makeRegistar()

mpl_plot_to_bytes([fig, format])

tprint(in_str)

xarray_to_bytes(ds[, format])

AFL.automation.shared.utilities.listify(obj)

AFL.automation.shared.utilities.tprint(in_str)

AFL.automation.shared.utilities.makeRegistar()

AFL.automation.shared.utilities.mpl_plot_to_bytes(fig=None, format='svg')
```

# AFL.automation.shared.widgetui

AFL.automation.shared.utilities.xarray\_to\_bytes(ds, format='svg')

# **Functions**

clear_output([wait])	Clear the output of the current cell receiving output.
<pre>client_construct_ui(self[, return_ui,])</pre>	
display(*objs[, include, exclude, metadata,])	Display a Python object in all frontends.

## **Classes**

```
Markdown([data, url, filename, metadata])
```

AFL.automation.shared.widgetui.client\_construct\_ui(self, return\_ui=False, display\_ui=True)

# 6.1.2 AFL.automation.sample

## **Modules**

```
CastingServer
```

# AFL.automation.sample.CastingServer

## **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x, l)	Return the square root of x.

#### **Classes**

```
Client([ip, port, username, interactive])

Driver(name[, defaults, overrides])

OT2Client([ip, port, username, interactive])

Communicate with APIServer

Communicate with AFL-automation server on OT-2
```

```
\textbf{class} \  \, \textbf{AFL}. automation.sample.CastingServer.\textbf{\textit{CastingServer}} (\textit{prep\_url}, \textit{overrides=None})
```

```
defaults = {'manifest': '/home/af1642/', 'manifest_cast': '/home/af1642/',
    'manifest_prep': '/home/af1642/'}

__init__(prep_url, overrides=None)
init_casting_manifest(attrs=None, overwrite=False)
status()
update_status(value)
log_event(manifest_key, event, write=True)
assign_targets(protocols, target_map)
prepare_casting_stocks(**spec)
    Spec should contain sample_name and a protocol
bulk_cast_films(**spec)
    Spec should contain sample_name and a protocol
prepare_and_cast(**sample)
```

# 6.2 Modules

This page provides an overview of the main modules in the AFL-automation framework.

# 6.2.1 API Server

The APIServer module provides HTTP microservices with authentication, task queueing, and UI generation.

AFL.automation.APIServer

# AFL.automation.APIServer

# **Modules**

APIServer	
Client	
Driver	
DummyDriver	
DummyOT2Driver	
LoggerFilter	
QueueDaemon	

## AFL.automation.APIServer.APIServer

#### **Functions**

<pre>create_access_token(identity[, fresh,])</pre>	Create a new access token.
<pre>create_refresh_token(identity[,])</pre>	Create a new refresh token.
<pre>get_jwt_identity()</pre>	In a protected endpoint, this will return the identity of the JWT that is accessing the endpoint.
<pre>jsonify(*args, **kwargs)</pre>	Serialize the given arguments as JSON, and return a Response object with the application/json mimetype.
<pre>jwt_required([optional, fresh, refresh,])</pre>	A decorator to protect a Flask endpoint with JSON Web Tokens.
listify(obj)	
<pre>render_template(template_name_or_list, **context)</pre>	Render a template by name with the given context.
<pre>send_file(path_or_file[, mimetype,])</pre>	Send the contents of a file to the client.
set_access_cookies(response,[, max_age,])	Modifiy a Flask Response to set a cookie containing the access JWT.
set_refresh_cookies(response,[,])	Modifiy a Flask Response to set a cookie containing the refresh JWT.
strtobool(val)	Convert a string representation of truth to true (1) or false (0).
<pre>unset_jwt_cookies(response[, domain])</pre>	Modifiy a Flask Response to delete the cookies containing access or refresh JWTs.

## AFL.automation.APIServer.APIServer.create\_access\_token

 $\begin{tabular}{ll} AFL. automation. APIServer. APIServer. create\_access\_token (identity: Any, fresh: bool | float | timedelta = False, expires\_delta: Literal[False] | timedelta | None = None, additional\_claims=None, additional\_headers=None) \end{tabular}$ 

Create a new access token.

## **Parameters**

- **identity** The identity of this token. This must either be a string, or you must have defined user\_identity\_loader() in order to convert the object you passed in into a string.
- **fresh** If this token should be marked as fresh, and can thus access endpoints protected with @jwt\_required(fresh=True). Defaults to False.

This value can also be a datetime.timedelta, which indicate how long this token will be considered fresh.

- expires\_delta A datetime.timedelta for how long this token should last before it expires. Set to False to disable expiration. If this is None, it will use the JWT\_ACCESS\_TOKEN\_EXPIRES config value (see Configuration Options)
- additional\_claims Optional. A hash of claims to include in the access token. These claims are merged into the default claims (exp, iat, etc) and claims returned from the additional\_claims\_loader() callback. On conflict, these claims take precedence.

• headers — Optional. A hash of headers to include in the access token. These headers are merged into the default headers (alg, typ) and headers returned from the additional\_headers\_loader() callback. On conflict, these headers take precedence.

#### Returns

An encoded access token

# AFL.automation.APIServer.APIServer.create\_refresh\_token

AFL.automation.APIServer.APIServer.create\_refresh\_token(identity: Any, expires\_delta: Literal[False] | timedelta | None = None, additional\_claims=None, additional\_headers=None)

Create a new refresh token.

#### **Parameters**

- **identity** The identity of this token. This must either be a string, or you must have defined user\_identity\_loader() in order to convert the object you passed in into a string.
- expires\_delta A datetime.timedelta for how long this token should last before it expires. Set to False to disable expiration. If this is None, it will use the JWT\_REFRESH\_TOKEN\_EXPIRES config value (see Configuration Options)
- additional\_claims Optional. A hash of claims to include in the refresh token. These claims are merged into the default claims (exp, iat, etc) and claims returned from the additional\_claims\_loader() callback. On conflict, these claims take precedence.
- headers Optional. A hash of headers to include in the refresh token. These headers are merged into the default headers (alg, typ) and headers returned from the additional\_headers\_loader() callback. On conflict, these headers take precedence.

## Returns

An encoded refresh token

## AFL.automation.APIServer.APIServer.get jwt identity

AFL.automation.APIServer.APIServer.get\_jwt\_identity() → Any

In a protected endpoint, this will return the identity of the JWT that is accessing the endpoint. If no JWT is present due to jwt\_required(optional=True), None is returned.

#### Returns

The identity of the JWT in the current request

## AFL.automation.APIServer.APIServer.jsonify

AFL.automation.APIServer.APIServer.jsonify(\*args: t.Any, \*\*kwargs: t.Any)  $\rightarrow$  Response

Serialize the given arguments as JSON, and return a Response object with the application/json mimetype. A dict or list returned from a view will be converted to a JSON response automatically without needing to call this.

This requires an active request or application context, and calls app.json.response().

In debug mode, the output is formatted with indentation to make it easier to read. This may also be controlled by the provider.

Either positional or keyword arguments can be given, not both. If no arguments are given, None is serialized.

#### **Parameters**

- args A single value to serialize, or multiple values to treat as a list to serialize.
- **kwargs** Treat as a dict to serialize.

Changed in version 2.2: Calls current\_app.json.response, allowing an app to override the behavior.

Changed in version 2.0.2: decimal. Decimal is supported by converting to a string.

Changed in version 0.11: Added support for serializing top-level arrays. This was a security risk in ancient browsers. See security-json.

Added in version 0.2.

## AFL.automation.APIServer.APIServer.jwt required

```
AFL.automation.APIServer.jwt_required(optional: bool = False, fresh: bool = False, refresh: bool = False, locations: str \mid Sequence \mid None = None, verify\_type: bool = True, skip\_revocation\_check: bool = False) <math>\rightarrow Any
```

A decorator to protect a Flask endpoint with JSON Web Tokens.

Any route decorated with this will require a valid JWT to be present in the request (unless optional=True, in which case no JWT is also valid) before the endpoint can be called.

#### **Parameters**

- **optional** If True, allow the decorated endpoint to be accessed if no JWT is present in the request. Defaults to False.
- fresh If True, require a JWT marked with fresh to be able to access this endpoint.
   Defaults to False.
- **refresh** If True, requires a refresh JWT to access this endpoint. If False, requires an access JWT to access this endpoint. Defaults to False.
- locations A location or list of locations to look for the JWT in this request, for example 'headers' or ['headers', 'cookies']. Defaults to None which indicates that JWTs will be looked for in the locations defined by the JWT\_TOKEN\_LOCATION configuration option.
- **verify\_type** If True, the token type (access or refresh) will be checked according to the refresh argument. If False, type will not be checked and both access and refresh tokens will be accepted.
- **skip\_revocation\_check** If True, revocation status of the token will be *not* checked. If False, revocation status of the token will be checked.

## AFL.automation.APIServer.APIServer.listify

AFL.automation.APIServer.APIServer.listify(obj)

## AFL.automation.APIServer.APIServer.render\_template

```
AFL.automation.APIServer.APIServer.render_template(template_name_or_list: str | Template | List[str | Template], **context: Any) \rightarrow str
```

Render a template by name with the given context.

#### **Parameters**

- **template\_name\_or\_list** The name of the template to render. If a list is given, the first name to exist will be rendered.
- **context** The variables to make available in the template.

## AFL.automation.APIServer.APIServer.send file

```
AFL.automation.APIServer.APIServer.send_file(path\_or\_file: PathLike \mid str \mid BinaryIO, mimetype: str \mid None = None, as\_attachment: bool = False, download\_name: str \mid None = None, conditional: bool = True, etag: bool | str = True, last\_modified: datetime | int | float | None = None, max\_age: int | Callable[[str | None], int | None] | None = None) <math>\rightarrow Response
```

Send the contents of a file to the client.

The first argument can be a file path or a file-like object. Paths are preferred in most cases because Werkzeug can manage the file and get extra information from the path. Passing a file-like object requires that the file is opened in binary mode, and is mostly useful when building a file in memory with io.BytesIO.

Never pass file paths provided by a user. The path is assumed to be trusted, so a user could craft a path to access a file you didn't intend. Use send\_from\_directory() to safely serve user-requested paths from within a directory.

If the WSGI server sets a file\_wrapper in environ, it is used, otherwise Werkzeug's built-in wrapper is used. Alternatively, if the HTTP server supports X-Sendfile, configuring Flask with USE\_X\_SENDFILE = True will tell the server to send the given path, which is much more efficient than reading it in Python.

#### **Parameters**

- path\_or\_file The path to the file to send, relative to the current working directory if a relative path is given. Alternatively, a file-like object opened in binary mode. Make sure the file pointer is seeked to the start of the data.
- **mimetype** The MIME type to send for the file. If not provided, it will try to detect it from the file name.
- **as\_attachment** Indicate to a browser that it should offer to save the file instead of displaying it.
- **download\_name** The default name browsers will use when saving the file. Defaults to the passed file name.
- **conditional** Enable conditional and range responses based on request headers. Requires passing a file path and **environ**.

- etag Calculate an ETag for the file, which requires passing a file path. Can also be a string to use instead.
- last\_modified The last modified time to send for the file, in seconds. If not provided, it will try to detect it from the file path.
- max\_age How long the client should cache the file, in seconds. If set, Cache-Control will be public, otherwise it will be no-cache to prefer conditional caching.

Changed in version 2.0: download\_name replaces the attachment\_filename parameter. If as\_attachment=False, it is passed with Content-Disposition: inline instead.

Changed in version 2.0: max\_age replaces the cache\_timeout parameter. conditional is enabled and max\_age is not set by default.

Changed in version 2.0: etag replaces the add\_etags parameter. It can be a string to use instead of generating one.

Changed in version 2.0: Passing a file-like object that inherits from TextIOBase will raise a ValueError rather than sending an empty file.

Added in version 2.0: Moved the implementation to Werkzeug. This is now a wrapper to pass some Flask-specific arguments.

Changed in version 1.1: filename may be a PathLike object.

Changed in version 1.1: Passing a BytesIO object supports range requests.

Changed in version 1.0.3: Filenames are encoded with ASCII instead of Latin-1 for broader compatibility with WSGI servers.

Changed in version 1.0: UTF-8 filenames as specified in RFC 2231 are supported.

Changed in version 0.12: The filename is no longer automatically inferred from file objects. If you want to use automatic MIME and etag support, pass a filename via filename\_or\_fp or attachment\_filename.

Changed in version 0.12: attachment\_filename is preferred over filename for MIME detection.

Changed in version 0.9: cache\_timeout defaults to Flask.get\_send\_file\_max\_age().

Changed in version 0.7: MIME guessing and etag support for file-like objects was deprecated because it was unreliable. Pass a filename if you are able to, otherwise attach an etag yourself.

Changed in version 0.5: The add\_etags, cache\_timeout and conditional parameters were added. The default behavior is to add etags.

Added in version 0.2.

# AFL.automation.APIServer.APIServer.set access cookies

AFL.automation.APIServer.APIServer.set\_access\_cookies(response: Response, encoded\_access\_token:  $str, max\_age=None, domain=None) \rightarrow None$ 

Modifiy a Flask Response to set a cookie containing the access JWT. Also sets the corresponding CSRF cookies if JWT\_CSRF\_IN\_COOKIES is True (see Configuration Options)

#### **Parameters**

- **response** A Flask Response object.
- **encoded\_access\_token** The encoded access token to set in the cookies.

- max\_age The max age of the cookie. If this is None, it will use the JWT\_SESSION\_COOKIE option (see Configuration Options). Otherwise, it will use this as the cookies max-age and the JWT\_SESSION\_COOKIE option will be ignored. Values should be the number of seconds (as an integer).
- **domain** The domain of the cookie. If this is None, it will use the JWT\_COOKIE\_DOMAIN option (see Configuration Options). Otherwise, it will use this as the cookies domain and the JWT\_COOKIE\_DOMAIN option will be ignored.

# AFL.automation.APIServer.APIServer.set refresh cookies

```
AFL.automation.APIServer.APIServer.set_refresh_cookies(response: Response, encoded_refresh_token: str, max\_age: int \mid None = None, domain: str \mid None = None) \rightarrow None
```

Modifiy a Flask Response to set a cookie containing the refresh JWT. Also sets the corresponding CSRF cookies if JWT\_CSRF\_IN\_COOKIES is True (see Configuration Options)

#### **Parameters**

- **response** A Flask Response object.
- **encoded\_refresh\_token** The encoded refresh token to set in the cookies.
- max\_age The max age of the cookie. If this is None, it will use the JWT\_SESSION\_COOKIE option (see Configuration Options). Otherwise, it will use this as the cookies max-age and the JWT\_SESSION\_COOKIE option will be ignored. Values should be the number of seconds (as an integer).
- **domain** The domain of the cookie. If this is None, it will use the JWT\_COOKIE\_DOMAIN option (see Configuration Options). Otherwise, it will use this as the cookies **domain** and the JWT\_COOKIE\_DOMAIN option will be ignored.

#### AFL.automation.APIServer.APIServer.strtobool

AFL.automation.APIServer.APIServer.strtobool(val)

Convert a string representation of truth to true (1) or false (0).

True values are 'y', 'yes', 't', 'true', 'on', and '1'; false values are 'n', 'no', 'f', 'false', 'off', and '0'. Raises ValueError if 'val' is anything else.

#### AFL.automation.APIServer.APIServer.unset jwt cookies

```
AFL.automation.APIServer.APIServer.unset_jwt_cookies(response: Response, domain: str \mid None = None \rightarrow None
```

Modifiy a Flask Response to delete the cookies containing access or refresh JWTs. Also deletes the corresponding CSRF cookies if applicable.

## **Parameters**

response – A Flask Response object

# Classes

APIServer(name[, data, experiment, contact,])	
CORS([app])	Initializes Cross Origin Resource sharing for the application.
FileHandler(filename[, mode, encoding,])	A handler class which writes formatted logging records to disk files.
Flask(import_name[, static_url_path,])	The flask object implements a WSGI application and acts as the central object.
<pre>IPVersion(value[, names, module, qualname,])</pre>	
<pre>JWTManager([app, add_context_processor])</pre>	An object used to hold JWT settings and callback functions for the Flask-JWT-Extended extension.
LoggerFilter(*filters)	
MutableQueue()	Thread-safe, mutable queue
QueueDaemon(app, driver, task_queue, history)	
SMTPHandler(mailhost, fromaddr, toaddrs, subject)	A handler class which sends an SMTP email for each logging event.
ServiceInfo	Service information.
Zeroconf([interfaces, unicast, ip_version,])	Implementation of Zeroconf Multicast DNS Service Discovery

## AFL.automation.APIServer.APIServer.APIServer

\_\_init\_\_(name, data=None, experiment='Development', contact='tbm@nist.gov', index\_template='index.html', new\_index\_template='index-new.html', plot\_template='simple-bokeh.html')

## **Methods**

```
__init__(name[, data, experiment, contact, ...])

add_standard_routes()

add_unqueued_routes()

advertise_zeroconf(**kwargs)

continues on next page
```

Table 1 – continued from previous page

Table 1 – continued	rom previous page
<pre>clear_history()</pre>	
<pre>clear_queue()</pre>	
<pre>create_queue(driver[, add_unqueued])</pre>	
debug()	
deposit_obj()	Store an object named obj in the driver's dropbox If a uuid is provided, the object will be stored with that uuid Otherwise, a new uuid will be generated.
driver_status()	_
enqueue()	
<pre>get_driver_object()</pre>	
<pre>get_info()</pre>	Live, status page of the robot
get_queue()	1 8
<pre>get_queue_iteration()</pre>	
<pre>get_queued_commands()</pre>	
<pre>get_quickbar()</pre>	Return the functions, params, and defaults to be shown in this server's quickbar
<pre>get_server_time()</pre>	
<pre>get_unqueued_commands()</pre>	
halt()	
index()	Live, status page of the robot
<pre>index_new()</pre>	Live, status page of the robot
<pre>init()</pre>	
<pre>init_logging([toaddrs])</pre>	
<pre>is_server_live()</pre>	
login()	
login_test()	
<pre>move_item()</pre>	
pause()	
query_driver()	
<pre>queue_state()</pre>	
	continues on next page

continues on next page

Table 1 – continued from previous page

```
remove_item()
 remove_items()
 render_unqueued(func, kwargs_add, **kwargs)
                                                   Convert an unqueued return item into web-suitable
                                                   output
 reorder_queue()
 reset_queue_daemon([driver])
 retrieve_obj()
                                                   Retrieve an object from the driver's dropbox uuid
                                                   specifies the object to retrieve delete specifies
                                                   whether to delete the object after retrieval
 run(**kwargs)
 run_threaded([start thread])
 send_1d_plot(result[, multi])
 send_array_as_jpg(array[, log_image, ...])
 set_driver_object()
 webapp()
                                                   Live, status page of the robot
__init__(name, data=None, experiment='Development', contact='tbm@nist.gov',
          index_template='index.html', new_index_template='index-new.html',
          plot_template='simple-bokeh.html')
create_queue(driver, add_unqueued=True)
reset_queue_daemon(driver=None)
advertise_zeroconf(**kwargs)
run(**kwargs)
run_threaded(start_thread=True, **kwargs)
add_standard_routes()
get_info()
    Live, status page of the robot
get_quickbar()
     Return the functions, params, and defaults to be shown in this server's quickbar
is_server_live()
get_unqueued_commands()
get_queued_commands()
add_unqueued_routes()
```

```
query_driver()
init_logging(toaddrs=None)
index()
     Live, status page of the robot
index_new()
     Live, status page of the robot
webapp()
     Live, status page of the robot
render_unqueued(func, kwargs_add, **kwargs)
     Convert an unqueued return item into web-suitable output
send_1d_plot(result, multi=False, **kwargs)
send_array_as_jpg(array, log_image=False, max_val=None, fillna=0.0, **kwargs)
queue_state()
driver_status()
get_queue()
get_queue_iteration()
deposit_obj()
     Store an object named obj in the driver's dropbox If a uuid is provided, the object will be stored with that
     uuid Otherwise, a new uuid will be generated. In either case, the uuid will be returned to the client.
retrieve_obj()
     Retrieve an object from the driver's dropbox unid specifies the object to retrieve delete specifies whether
     to delete the object after retrieval
set_driver_object()
get_driver_object()
enqueue()
reorder_queue()
remove_items()
remove_item()
move_item()
clear_queue()
clear_history()
debug()
pause()
halt()
```

```
init()
login()
get_server_time()
login_test()
```

## AFL.automation.APIServer.APIServer.CORS

class AFL.automation.APIServer.APIServer.CORS(app=None, \*\*kwargs)

Bases: object

Initializes Cross Origin Resource sharing for the application. The arguments are identical to *cross\_origin*, with the addition of a *resources* parameter. The resources parameter defines a series of regular expressions for resource paths to match and optionally, the associated options to be applied to the particular resource. These options are identical to the arguments to *cross\_origin*.

The settings for CORS are determined in the following order

- 1. Resource level settings (e.g when passed as a dictionary)
- 2. Keyword argument settings
- 3. App level configuration settings (e.g. CORS\_\*)
- 4. Default settings

Note: as it is possible for multiple regular expressions to match a resource path, the regular expressions are first sorted by length, from longest to shortest, in order to attempt to match the most specific regular expression. This allows the definition of a number of specific resource options, with a wildcard fallback for all other resources.

#### **Parameters**

• **resources** (*dict*, *iterable or string*) – The series of regular expression and (optionally) associated CORS options to be applied to the given resource path.

If the argument is a dictionary, it's keys must be regular expressions, and the values must be a dictionary of kwargs, identical to the kwargs of this function.

If the argument is a list, it is expected to be a list of regular expressions, for which the appwide configured options are applied.

If the argument is a string, it is expected to be a regular expression for which the app-wide configured options are applied.

Default: Match all and apply app-level configuration

• **origins** (*list*, *string or regex*) – The origin, or list of origins to allow requests from. The origin(s) may be regular expressions, case-sensitive strings, or else an asterisk.



origins must include the schema and the port (if not port 80), e.g., CORS(app, origins=["http://localhost:8000", "https://example.com"]).

Default: '\*'

• **methods** (*list or string*) – The method or list of methods which the allowed origins are allowed to access for non-simple requests.

Default: [GET, HEAD, POST, OPTIONS, PUT, PATCH, DELETE]

• **expose\_headers** (*list or string*) – The header or list which are safe to expose to the API of a CORS API specification.

Default: None

• allow\_headers (list, string or regex) — The header or list of header field names which can be used when this resource is accessed by allowed origins. The header(s) may be regular expressions, case-sensitive strings, or else an asterisk.

Default: '\*', allow all headers

• **supports\_credentials** (*bool*) – Allows users to make authenticated requests. If true, injects the *Access-Control-Allow-Credentials* header in responses. This allows cookies and credentials to be submitted across domains.

#### note

This option cannot be used in conjunction with a '\*' origin

Default: False

• max\_age (timedelta, integer, string or None) — The maximum time for which this CORS request maybe cached. This value is set as the *Access-Control-Max-Age* header.

Default: None

• **send\_wildcard** (*bool*) – If True, and the origins parameter is \*, a wildcard *Access-Control-Allow-Origin* header is sent, rather than the request's *Origin* header.

Default: False

implementation guidelines.

• vary\_header (bool) - If True, the header Vary: Origin will be returned as per the W3

Setting this header when the *Access-Control-Allow-Origin* is dynamically generated (e.g. when there is more than one allowed origin, and an Origin than '\*' is returned) informs CDNs and other caches that the CORS headers are dynamic, and cannot be cached.

If False, the Vary header will never be injected or altered.

Default: True

• allow\_private\_network (bool) — If True, the response header Access-Control-Allow-Private-Network will be set with the value 'true' whenever the request header Access-Control-Request-Private-Network has a value 'true'.

If False, the response header *Access-Control-Allow-Private-Network* will be set with the value 'false' whenever the request header *Access-Control-Request-Private-Network* has a value of 'true'.

If the request header *Access-Control-Request-Private-Network* is not present or has a value other than 'true', the response header *Access-Control-Allow-Private-Network* will not be set.

Default: True

\_\_init\_\_(app=None, \*\*kwargs)

# **Methods**

```
__init__([app])
init_app(app, **kwargs)

__init__(app=None, **kwargs)
init_app(app, **kwargs)
```

# AFL.automation.APIServer.APIServer.FileHandler

Bases: StreamHandler

A handler class which writes formatted logging records to disk files.

 $\verb|\__init__(filename, mode='a', encoding=None, delay=False, errors=None)|$ 

Open the specified file and use it as the stream for logging.

## **Methods**

init(filename[, mode, encoding, delay,])	Open the specified file and use it as the stream for logging.
acquire()	Acquire the I/O thread lock.
addFilter(filter)	Add the specified filter to this handler.
close()	Closes the stream.
createLock()	Acquire a thread lock for serializing access to the underlying I/O.
emit(record)	Emit a record.
filter(record)	Determine if a record is loggable by consulting all the filters.
flush()	Flushes the stream.
<pre>format(record)</pre>	Format the specified record.
<pre>get_name()</pre>	
handle(record)	Conditionally emit the specified logging record.
handleError(record)	Handle errors which occur during an emit() call.
release()	Release the I/O thread lock.
removeFilter(filter)	Remove the specified filter from this handler.
setFormatter(fmt)	Set the formatter for this handler.
setLevel(level)	Set the logging level of this handler.
setStream(stream)	Sets the StreamHandler's stream to the specified value, if it is different.
set_name(name)	

#### **Attributes**

#### name

#### terminator

\_\_init\_\_(filename, mode='a', encoding=None, delay=False, errors=None)

Open the specified file and use it as the stream for logging.

#### close()

Closes the stream.

#### emit(record)

Emit a record.

If the stream was not opened because 'delay' was specified in the constructor, open it before calling the superclass's emit.

If stream is not open, current mode is 'w' and \_closed=True, record will not be emitted (see Issue #42378).

## acquire()

Acquire the I/O thread lock.

#### addFilter(filter)

Add the specified filter to this handler.

#### createLock()

Acquire a thread lock for serializing access to the underlying I/O.

## filter(record)

Determine if a record is loggable by consulting all the filters.

The default is to allow the record to be logged; any filter can veto this and the record is then dropped. Returns a zero value if a record is to be dropped, else non-zero.

Changed in version 3.2: Allow filters to be just callables.

## flush()

Flushes the stream.

## format(record)

Format the specified record.

If a formatter is set, use it. Otherwise, use the default formatter for the module.

#### get\_name()

# handle(record)

Conditionally emit the specified logging record.

Emission depends on filters which may have been added to the handler. Wrap the actual emission of the record with acquisition/release of the I/O thread lock. Returns whether the filter passed the record for emission.

## handleError(record)

Handle errors which occur during an emit() call.

This method should be called from handlers when an exception is encountered during an emit() call. If raiseExceptions is false, exceptions get silently ignored. This is what is mostly wanted for a logging system - most users will not care about errors in the logging system, they are more interested in application errors. You could, however, replace this with a custom handler if you wish. The record which was being processed is passed in to this method.

```
release()
    Release the I/O thread lock.

removeFilter(filter)
    Remove the specified filter from this handler.

setFormatter(fint)
    Set the formatter for this handler.

setLevel(level)
    Set the logging level of this handler. level must be an int or a str.

setStream(stream)
    Sets the StreamHandler's stream to the specified value, if it is different.
    Returns the old stream, if the stream was changed, or None if it wasn't.

set_name(name)

terminator = '\n'
```

#### AFL.automation.APIServer.APIServer.Flask

```
class AFL.automation.APIServer.APIServer.Flask(import_name: str, static_url_path: str | None = None, static_folder: str | PathLike | None = 'static', static_host: str | None = None, host_matching: bool = False, subdomain_matching: bool = False, template_folder: str | PathLike | None = 'templates', instance_path: str | None = None, instance_relative_config: bool = False, root_path: str | None = None)
```

Bases: Scaffold

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an <code>\_\_init\_\_.py</code> file inside) or a standard module (just a .py file).

For more information about resource loading, see open\_resource().

Usually you create a *Flask* instance in your main module or in the \_\_init\_\_.py file of your package like this:

```
from flask import Flask
app = Flask(__name__)
```

# **1** About the First Parameter

The idea of the first parameter is to give Flask an idea of what belongs to your application. This name is used to find resources on the filesystem, can be used by extensions to improve debugging information and a lot more.

So it's important what you provide there. If you are using a single module, \_\_name\_\_ is always the correct value. If you however are using a package, it's usually recommended to hardcode the name of your package there.

For example if your application is defined in yourapplication/app.py you should create it with one of the two versions below:

```
app = Flask('yourapplication')
app = Flask(__name__.split('.')[0])
```

Why is that? The application will work even with \_\_name\_\_, thanks to how resources are looked up. However it will make debugging more painful. Certain extensions can make assumptions based on the import name of your application. For example the Flask-SQLAlchemy extension will look for the code in your application that triggered an SQL query in debug mode. If the import name is not properly set up, that debugging information is lost. (For example it would only pick up SQL queries in *yourapplication.app* and not *yourapplication.views.frontend*)

Added in version 0.7: The static\_url\_path, static\_folder, and template\_folder parameters were added.

Added in version 0.8: The *instance\_path* and *instance\_relative\_config* parameters were added.

Added in version 0.11: The *root\_path* parameter was added.

Added in version 1.0: The host\_matching and static\_host parameters were added.

Added in version 1.0: The subdomain\_matching parameter was added. Subdomain matching needs to be enabled manually now. Setting SERVER\_NAME does not implicitly enable it.

#### **Parameters**

- **import\_name** the name of the application package
- **static\_url\_path** can be used to specify a different path for the static files on the web. Defaults to the name of the *static\_folder* folder.
- **static\_folder** The folder with static files that is served at **static\_url\_path**. Relative to the application **root\_path** or an absolute path. Defaults to 'static'.
- **static\_host** the host to use when adding the static route. Defaults to None. Required when using host\_matching=True with a **static\_folder** configured.
- host\_matching set url\_map.host\_matching attribute. Defaults to False.
- **subdomain\_matching** consider the subdomain relative to SERVER\_NAME when matching routes. Defaults to False.
- **template\_folder** the folder that contains the templates that should be used by the application. Defaults to 'templates' folder in the root path of the application.
- **instance\_path** An alternative instance path for the application. By default the folder 'instance' next to the package or module is assumed to be the instance path.
- **instance\_relative\_config** if set to True relative filenames for loading the config are assumed to be relative to the instance path instead of the application root.

• **root\_path** – The path to the root of the application files. This should only be set manually when it can't be detected automatically, such as for namespace packages.

```
__init__(import_name: str, static_url_path: str | None = None, static_folder: str | PathLike | None = 'static', static_host: str | None = None, host_matching: bool = False, subdomain_matching: bool = False, template_folder: str | PathLike | None = 'templates', instance_path: str | None = None, instance_relative_config: bool = False, root_path: str | None = None)
```

#### **Methods**

init(import_name[, static_url_path,])	
<pre>add_template_filter(f[, name])</pre>	Register a custom template filter.
add_template_global(f[, name])	Register a custom template global function.
add_template_test(f[, name])	Register a custom template test.
add_url_rule(rule[, endpoint, view_func,])	Register a rule for routing incoming requests and building URLs.
after_request(f)	Register a function to run after each request to this object.
<pre>app_context()</pre>	Create an AppContext.
async_to_sync(func)	Return a sync function that will run the coroutine function.
<pre>auto_find_instance_path()</pre>	Tries to locate the instance path if it was not provided to the constructor of the application class.
before_first_request(f)	Registers a function to be run before the first request to this instance of the application.
before_request(f)	Register a function to run before each request.
context_processor(f)	Registers a template context processor function.
<pre>create_global_jinja_loader()</pre>	Creates the loader for the Jinja2 environment.
<pre>create_jinja_environment()</pre>	Create the Jinja environment based on <i>jinja_options</i> and the various Jinja-related methods of the app.
<pre>create_url_adapter(request)</pre>	Creates a URL adapter for the given request.
delete(rule, **options)	Shortcut for <i>route()</i> with methods=["DELETE"].
<pre>dispatch_request()</pre>	Does the request dispatching.
<pre>do_teardown_appcontext([exc])</pre>	Called right before the application context is popped.
<pre>do_teardown_request([exc])</pre>	Called after the request is dispatched and the response is returned, right before the request context is popped.
<pre>endpoint(endpoint)</pre>	Decorate a view function to register it for the given endpoint.
ensure_sync(func)	Ensure that the function is synchronous for WSGI workers.
errorhandler(code_or_exception)	Register a function to handle errors by code or exception class.
<pre>finalize_request(rv[, from_error_handler])</pre>	Given the return value from a view function this finalizes the request by converting it into a response and invoking the postprocessing functions.
<pre>full_dispatch_request()</pre>	Dispatches the request and on top of that performs request pre and postprocessing as well as HTTP exception catching and error handling.
<pre>get(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["GET"].
	continues on next page

continues on next page

Table 2 – continued from previous page

get_send_file_max_age(filename)         Used by send_file() to determine the max_age cache value for a given file path if it wasn't passed.           handle_exception(e)         Handle an exception that did not have an error handler associated with it, or that was raised from an error handler.           handle_http_exception(e)         Handles an HTTP exception.           handle_usrl_build_error(error, endpoint, values)         Called by url_for() if a BuildError was raised.           inject_url_defaults(endpoint, values)         This method is called whenever an exception occurs that should be handled.           inject_blueprints()         Logs an exception.           iter_blueprints()         Logs an exception.           log_exception(exc_info)         Logs an exception.           make_aborter()         Create the object to assign to aborter.           make_offault_options_response()         Create the object to assign to aborter.           make_response(rv)         Create the object to assign to aborter.           make_response(rv)         Convert the return value from a view function to an instance of response_class.           make_shell_context()         Returns the shell context for an interactive shell for this application.           open_resource(resource[, mode])         Open a resource file relative to root_path for reading.           process_request()         Open a resource file relative to root_path for reading.           process_request()         Call	Table 2 – continued	i nom previous page
Handle_nexception(e)	<pre>get_send_file_max_age(filename)</pre>	
handle_url_build_error(error, endpoint, values)         Called by url_for() if a BuildError was raised.           handle_user_exception(e)         This method is called whenever an exception occurs that should be handled.           inject_url_defaults(endpoint, values)         Injects the URL defaults for the given endpoint directly into the values dictionary passed.           lerates over all blueprints by the order they were registered.         Logs an exception.           log_exception(exc_info)         Logs an exception.           make_aborter()         Used to create the config attribute by the Flask constructor.           make_default_options_response()         This method is called to create the default OPTIONS response.           make_response(rv)         Convert the return value from a view function to an instance of response_class.           make_shell_context()         Returns the shell context for an interactive shell for this application.           open_instance_resource(resource[, mode])         Opens a resource file relative to root_path for reading.           open_finstance_path()         Open a resource file relative to root_path for reading.           patch(rule, **options)         Shortcut for route() with methods=["PATCH"].           post(rule, **options)         Shortcut for route() with methods=["PUT"].           process_response(response)         Called before the request is dispatched.           profitered(location[, code])         Called before the request is dispatched. <td>handle_exception(e)</td> <td>Handle an exception that did not have an error handler associated with it, or that was raised from an error</td>	handle_exception(e)	Handle an exception that did not have an error handler associated with it, or that was raised from an error
handle_url_build_error(error, endpoint, values)         Called by url_for() if a BuildError was raised.           handle_user_exception(e)         This method is called whenever an exception occurs that should be handled.           inject_url_defaults(endpoint, values)         Injects the URL defaults for the given endpoint directly into the values dictionary passed.           lerates over all blueprints by the order they were registered.         Logs an exception.           log_exception(exc_info)         Logs an exception.           make_aborter()         Used to create the config attribute by the Flask constructor.           make_default_options_response()         This method is called to create the default OPTIONS response.           make_response(rv)         Convert the return value from a view function to an instance of response_class.           make_shell_context()         Returns the shell context for an interactive shell for this application.           open_instance_resource(resource[, mode])         Opens a resource file relative to root_path for reading.           open_finstance_path()         Open a resource file relative to root_path for reading.           patch(rule, **options)         Shortcut for route() with methods=["PATCH"].           post(rule, **options)         Shortcut for route() with methods=["PUT"].           process_response(response)         Called before the request is dispatched.           profitered(location[, code])         Called before the request is dispatched. <td>handle_http_exception(e)</td> <td>Handles an HTTP exception.</td>	handle_http_exception(e)	Handles an HTTP exception.
that should be handled.  Inject_url_defaults(endpoint, values)  Inject_url_defaults(endpoint, values)  Inject_url_defaults for the given endpoint directly into the values dictionary passed.  Iterates over all blueprints by the order they were registered.  Log_san exception.  Logs an exception.  Logs an exception.  Create the object to assign to aborter.  Used to create the config attribute by the Flask constructor.  In its method is called to create the default OPTIONS response.  Make_shell_context()  This method is called to create the default OPTIONS response.  Make_shell_context()  This method is called to create the default OPTIONS response.  Make_shell_context()  Returns the shell context for an interactive shell for this application.  Open_instance_resource(resource[, mode])  Opens a resource from the application's instance folder (instance_path).  Open a resource fine relative to root_path for reading.  Patch(rule, **options)  Shortcut for route() with methods=["PATCH"].  Shortcut for route() with methods=["POST"].  Called before the request is dispatched.  Can be overridden in order to modify the response object before it's sent to the WSGI server.  Put(rule, **options)  Taise_routing_exception(request)  Intercept routing exceptions and possibly do something else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_blueprint(blueprint, **options)  Register a Blueprint on the application.  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Logs an exception.  Runs the application on a local development server.  Returns True if autoescaping should be active for the given template name.  The view function used to serve files from static_folder.  Shell_context_processor(f)	handle_url_build_error(error, endpoint, val-	<u> </u>
rectly into the values dictionary passed.  iter_blueprints()	handle_user_exception(e)	that should be handled.
istered. Logs an exception.  make_aborter()  make_config([instance_relative])  make_default_options_response()  make_default_options_response()  make_response(rv)  make_shell_context()  make_shell_context()  make_shell_context()  make_shell_context()  make_shell_context()  open_instance_resource(resource[, mode])  open_resource(resource[, mode])  patch(rule, **options)  preprocess_request()  process_response(response)  Called before the request is dispatched.  Process_response(response)  Called before	<pre>inject_url_defaults(endpoint, values)</pre>	rectly into the values dictionary passed.
make_aborter()         Create the object to assign to aborter.           make_config([instance_relative])         Used to create the config attribute by the Flask constructor.           make_default_options_response()         This method is called to create the default OPTIONS response.           make_response(rv)         Convert the return value from a view function to an instance of response_class.           make_shell_context()         Returns the shell context for an interactive shell for this application.           open_instance_resource(resource[, mode])         Open a resource file relative to root_path for reading.           open_resource(resource[, mode])         Open a resource file relative to root_path for reading.           patch(rule, **options)         Shortcut for route() with methods=["PATCH"].           post(rule, **options)         Shortcut for route() with methods=["POST"].           precess_request()         Called before the request is dispatched.           process_response(response)         Can be overridden in order to modify the response object before it's sent to the WSGI server.           put(rule, **options)         Shortcut for route() with methods=["PUT"].           raise_routing_exception(request)         Intercept routing exceptions and possibly do something else.           redirect(location[, code])         Create a redirect response object.           register_blueprint(blueprint, **options)         Register a Blueprint on the application.	<pre>iter_blueprints()</pre>	
make_config([instance_relative])         Used to create the config attribute by the Flask constructor.           make_default_options_response()         This method is called to create the default OPTIONS response.           make_response(rv)         Convert the return value from a view function to an instance of response_class.           make_shell_context()         Returns the shell context for an interactive shell for this application.           open_instance_resource(resource[, mode])         Open a resource from the application's instance folder (instance_path).           open_resource(resource[, mode])         Open a resource file relative to root_path for reading.           shortcut for route() with methods=["PATCH"].         Shortcut for route() with methods=["PATCH"].           post(rule, **options)         Shortcut for route() with methods=["POST"].           preprocess_request()         Called before the request is dispatched.           process_response(response)         Can be overridden in order to modify the response object before it's sent to the WSGI server.           put(rule, **options)         Shortcut for route() with methods=["PUT"].           raise_routing_exception(request)         Intercept routing exceptions and possibly do something else.           redirect(location[, code])         Create a Relueprint on the application.           register_blueprint(blueprint, **options)         Register a Blueprint on the application.           register_blueprint(blueprint, **options)	<pre>log_exception(exc_info)</pre>	
structor.  This method is called to create the default OPTIONS response.  Make_response(rv)  Convert the return value from a view function to an instance of response_class.  Make_shell_context()  Returns the shell context for an interactive shell for this application.  Open_instance_resource(resource[, mode])  Opens a resource from the application's instance folder (instance_path).  Open a resource file relative to root_path for reading.  Patch(rule, **options)  Patch(rule, **options)  Shortcut for route() with methods=["POST"].  Proprocess_request()  Process_response(response)  Called before the request is dispatched.  Process_response(response)  Can be overridden in order to modify the response object before it's sent to the WSGI server.  Put(rule, **options)  This recept routing exceptions and possibly do something else.  Redirect(location[, code])  Register_allueprint(blueprint, **options)  Register a Blueprint on the application.  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  Request_context (environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Runs the application on a local development server.  Returns True if autoescaping should be active for the given template name.  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)	V V	
response.  Convert the return value from a view function to an instance of response_class.  make_shell_context()  Returns the shell context for an interactive shell for this application.  open_instance_resource(resource[, mode])  Opens a resource from the application's instance folder (instance_path).  Open a resource file relative to root_path for reading.  patch(rule, **options)  patch(rule, **options)  post(rule, **options)  preprocess_request()  process_response(response)  Called before the request is dispatched.  process_response(response)  Can be overridden in order to modify the response object before it's sent to the WSGI server.  put(rule, **options)  raise_routing_exception(request)  Intercept routing exceptions and possibly do something else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_error_handler(code_or_exception, f)  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  run([host, port, debug, load_dotenv])  select_jinja_autoescape(filename)  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)  Registers a shell context processor function.	<pre>make_config([instance_relative])</pre>	· · · · · · · · · · · · · · · · · · ·
instance of response_class.  Returns the shell context for an interactive shell for this application.  open_instance_resource(resource[, mode])  Opens a resource from the application's instance folder (instance_path).  Open a resource file relative to root_path for reading.  patch(rule, **options)  post(rule, **options)  post(rule, **options)  post(rule, **options)  preprocess_request()  process_response(response)  Called before the request is dispatched.  Can be overridden in order to modify the response object before it's sent to the WSGI server.  put(rule, **options)  raise_routing_exception(request)  Intercept routing exceptions and possibly do something else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_error_handler(code_or_exception, f)  Alternative error attach function to the errorhandler(of decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Pecorate a view function to register it with the given URL rule and options.  run([host, port, debug, load_dotenv])  Runs the application on a local development server.  select_jinja_autoescape(filename)  Returns True if autoescaping should be active for the given template name.  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)	<pre>make_default_options_response()</pre>	
this application.  open_instance_resource(resource[, mode])  open_resource(resource[, mode])  open_resource(resource[, mode])  open_resource(resource[, mode])  open_resource(resource[, mode])  patch(rule, **options)  post(rule, **options)  post(rule, **options)  preprocess_request()  process_response(response)  Called before the request is dispatched.  process_response(response)  Can be overridden in order to modify the response object before it's sent to the WSGI server.  put(rule, **options)  raise_routing_exception(request)  Intercept routing exceptions and possibly do something else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_error_handler(code_or_exception, f)  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Decorate a view function to register it with the given URL rule and options.  run([host, port, debug, load_dotenv])  select_jinja_autoescape(filename)  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)  Registers a shell context processor function.	make_response(rv)	
folder (instance_path).  open_resource(resource[, mode])  Open a resource file relative to root_path for reading.  patch(rule, **options)	<pre>make_shell_context()</pre>	
patch(rule, **options) post(rule, **options) post(rule, **options) preprocess_request() process_response(response) Called before the request is dispatched. Can be overridden in order to modify the response object before it's sent to the WSGI server.  put(rule, **options) Shortcut for route() with methods=["PUT"].  raise_routing_exception(request) Intercept routing exceptions and possibly do something else.  redirect(location[, code]) register_blueprint(blueprint, **options) register_error_handler(code_or_exception, f) Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ) Create a RequestContext representing a WSGI environment.  route(rule, **options) Decorate a view function to register it with the given URL rule and options.  rum([host, port, debug, load_dotenv]) Runs the application on a local development server.  select_jinja_autoescape(filename) Returns True if autoescaping should be active for the given template name.  send_static_file(filename) The view function used to serve files from static_folder.  shell_context_processor(f) Registers a shell context processor function.	<pre>open_instance_resource(resource[, mode])</pre>	•
post(rule, **options) preprocess_request() process_response(response) Called before the request is dispatched. process_response(response) Can be overridden in order to modify the response object before it's sent to the WSGI server. put(rule, **options) Shortcut for route() with methods=["PUT"]. Intercept routing exceptions and possibly do something else. redirect(location[, code]) register_blueprint(blueprint, **options) register_error_handler(code_or_exception, f) Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage. request_context(environ) Create a RequestContext representing a WSGI environment. route(rule, **options) Decorate a view function to register it with the given URL rule and options. run([host, port, debug, load_dotenv]) Runs the application on a local development server. select_jinja_autoescape(filename) Returns True if autoescaping should be active for the given template name. send_static_file(filename) The view function used to serve files from static_folder. shell_context_processor(f) Registers a shell context processor function.	<pre>open_resource(resource[, mode])</pre>	<u>-</u>
preprocess_request()Called before the request is dispatched.process_response(response)Can be overridden in order to modify the response object before it's sent to the WSGI server.put(rule, **options)Shortcut for route() with methods=["PUT"].raise_routing_exception(request)Intercept routing exceptions and possibly do something else.redirect(location[, code])Create a redirect response object.register_blueprint(blueprint, **options)Register a Blueprint on the application.register_error_handler(code_or_exception, f)Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	<pre>patch(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["PATCH"].
process_response(response)Can be overridden in order to modify the response object before it's sent to the WSGI server.put(rule, **options)Shortcut for route() with methods=["PUT"].raise_routing_exception(request)Intercept routing exceptions and possibly do something else.redirect(location[, code])Create a redirect response object.register_blueprint(blueprint, **options)Register a Blueprint on the application.register_error_handler(code_or_exception, f)Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	<pre>post(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["POST"].
object before it's sent to the WSGI server.  put(rule, **options) Shortcut for route() with methods=["PUT"].  raise_routing_exception(request) Intercept routing exceptions and possibly do something else.  redirect(location[, code]) Create a redirect response object.  register_blueprint(blueprint, **options) Register a Blueprint on the application.  register_error_handler(code_or_exception, f) Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ) Create a RequestContext representing a WSGI environment.  route(rule, **options) Decorate a view function to register it with the given URL rule and options.  run([host, port, debug, load_dotenv]) Runs the application on a local development server.  select_jinja_autoescape(filename) Returns True if autoescaping should be active for the given template name.  send_static_file(filename) The view function used to serve files from static_folder.  shell_context_processor(f) Registers a shell context processor function.	<pre>preprocess_request()</pre>	
raise_routing_exception(request)  Intercept routing exceptions and possibly do something else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_error_handler(code_or_exception, f)  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Decorate a view function to register it with the given URL rule and options.  run([host, port, debug, load_dotenv])  Returns True if autoescaping should be active for the given template name.  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)  Registers a shell context processor function.	process_response(response)	
thing else.  redirect(location[, code])  register_blueprint(blueprint, **options)  register_error_handler(code_or_exception, f)  Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.  request_context(environ)  Create a RequestContext representing a WSGI environment.  route(rule, **options)  Decorate a view function to register it with the given URL rule and options.  run([host, port, debug, load_dotenv])  Returns True if autoescaping should be active for the given template name.  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)  Registers a shell context processor function.	<pre>put(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["PUT"].
register_blueprint(blueprint, **options)Register a Blueprint on the application.register_error_handler(code_or_exception, f)Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	<pre>raise_routing_exception(request)</pre>	
register_error_handler(code_or_exception, f)Alternative error attach function to the errorhandler() decorator that is more straightforward to use for non decorator usage.request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.		
errorhandler() decorator that is more straightforward to use for non decorator usage.request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.		
request_context(environ)Create a RequestContext representing a WSGI environment.route(rule, **options)Decorate a view function to register it with the given URL rule and options.run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	register_error_handler(code_or_exception, f)	errorhandler() decorator that is more straightfor-
URL rule and options.         run([host, port, debug, load_dotenv])       Runs the application on a local development server.         select_jinja_autoescape(filename)       Returns True if autoescaping should be active for the given template name.         send_static_file(filename)       The view function used to serve files from static_folder.         shell_context_processor(f)       Registers a shell context processor function.	request_context(environ)	Create a RequestContext representing a WSGI en-
run([host, port, debug, load_dotenv])Runs the application on a local development server.select_jinja_autoescape(filename)Returns True if autoescaping should be active for the given template name.send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	route(rule, **options)	· · · · · · · · · · · · · · · · · · ·
given template name.  send_static_file(filename)  The view function used to serve files from static_folder.  shell_context_processor(f)  Registers a shell context processor function.	<pre>run([host, port, debug, load_dotenv])</pre>	Runs the application on a local development server.
send_static_file(filename)The view function used to serve files from static_folder.shell_context_processor(f)Registers a shell context processor function.	select_jinja_autoescape(filename)	
shell_context_processor(f) Registers a shell context processor function.	<pre>send_static_file(filename)</pre>	The view function used to serve files from
	shell_context_processor(f)	

continues on next page

Table 2 – continued from previous page

should_ignore_error(error)	This is called to figure out if an error should be ignored or not as far as the teardown system is concerned.
teardown_appcontext(f)	Registers a function to be called when the application context is popped.
teardown_request(f)	Register a function to be called when the request context is popped.
<pre>template_filter([name])</pre>	A decorator that is used to register custom template filter. You can specify a name for the filter, otherwise the function name will be used. Example::.
<pre>template_global([name])</pre>	A decorator that is used to register a custom template global function. You can specify a name for the global function, otherwise the function name will be used. Example::.
<pre>template_test([name])</pre>	A decorator that is used to register custom template test. You can specify a name for the test, otherwise the function name will be used. Example::.
<pre>test_cli_runner(**kwargs)</pre>	Create a CLI runner for testing CLI commands.
<pre>test_client([use_cookies])</pre>	Creates a test client for this application.
test_request_context(*args, **kwargs)	Create a RequestContext for a WSGI environment created from the given values.
trap_http_exception(e)	Checks if an HTTP exception should be trapped or not.
<pre>update_template_context(context)</pre>	Update the template context with some commonly used variables.
url_defaults(f)	Callback function for URL defaults for all view functions of the application.
<pre>url_for(endpoint, *[, _anchor, _method,])</pre>	Generate a URL to the given endpoint with the given values.
url_value_preprocessor(f)	Register a URL value preprocessor function for all view functions in the application.
wsgi_app(environ, start_response)	The actual WSGI application. This is not implemented incall() so that middlewares can be applied without losing a reference to the app object. Instead of doing this::.

# **Attributes**

debug	Whether debug mode is enabled.
default_config	Default configuration parameters.
env	What environment the app is running in.
<pre>got_first_request</pre>	This attribute is set to True if the application started
	handling the first request.
has_static_folder	True if static_folder is set.
jinja_env	The Jinja environment used to load templates.
jinja_loader	The Jinja loader for this object's templates.
jinja_options	Options that are passed to the Jinja environment in
	<pre>create_jinja_environment().</pre>
json_decoder	The JSON decoder class to use.
json_encoder	The JSON encoder class to use.

continues on next page

Table 3 – continued from previous page

Table 3 – continued	d from previous page
logger	A standard Python Logger for the app, with the same name as <i>name</i> .
name	The name of the application.
permanent_session_lifetime	A timedelta which is used to set the expiration date
permanent_session_rriceime	of a permanent session.
propagate_exceptions	Returns the value of the PROPAGATE_EXCEPTIONS
propagate_exceptions	configuration value in case it's set, otherwise a sen-
	sible default is returned.
secret_key	If a secret key is set, cryptographic components can
	use this to sign cookies and other things.
send_file_max_age_default	The default value for max_age for send_file().
session_cookie_name	The name of the cookie set by the session interface.
session_interface	the session interface to use.
static_folder	The absolute path to the configured static folder.
static_url_path	The URL prefix that the static route will be accessible
	from.
templates_auto_reload	Reload templates when they are changed.
test_cli_runner_class	The CliRunner subclass, by de-
	<pre>fault FlaskCliRunner that is used by test_cli_runner().</pre>
test_client_class	The test_client() method creates an instance of
	this test client class.
testing	The testing flag.
use_x_sendfile	Enable this to use the X-Sendfile feature, assuming
	the server supports it, from send_file().
instance_path	Holds the path to the instance folder.
config	The configuration dictionary as Config.
aborter	An instance of aborter_class created by make_aborter().
json	Provides access to JSON methods.
url_build_error_handlers	A list of functions that are called by handle_url_build_error() when url_for() raises a BuildError.
before_first_request_funcs	A list of functions that will be called at the beginning of the first request to this instance.
teardown_appcontext_funcs	A list of functions that are called when the application context is destroyed.
shell_context_processors	A list of shell context processor functions that should be run when a shell context is created.
blueprints	Maps registered blueprint names to blueprint objects.
extensions	a place where extensions can store application spe-
	cific state.
url_map	The Map for this instance. You can use this to change
	the routing converters after the class was created but
	before any routes are connected. Example::.
import_name	The name of the package or module that this object belongs to.
template_folder	The path to the templates folder, relative to $root_path$ , to add to the template loader.
root_path	Absolute path to the package on the filesystem.
cli	The Click command group for registering CLI com-
	mands for this object.
	continues on payt page

continues on next page

Table 3 – continued from previous page

view_functions	A dictionary mapping endpoint names to view functions.
error_handler_spec	A data structure of registered error handlers, in the format {scope: {code: {class: handler}}}.
before_request_funcs	A data structure of functions to call at the beginning of each request, in the format {scope: [functions]}.
after_request_funcs	A data structure of functions to call at the end of each request, in the format {scope: [functions]}.
teardown_request_funcs	A data structure of functions to call at the end of each request even if an exception is raised, in the format {scope: [functions]}.
template_context_processors	A data structure of functions to call to pass extra context values when rendering templates, in the format {scope: [functions]}.
url_value_preprocessors	A data structure of functions to call to modify the keyword arguments passed to the view function, in the format {scope: [functions]}.
url_default_functions	A data structure of functions to call to modify the keyword arguments when generating URLs, in the format {scope: [functions]}.

# request\_class

alias of Request

# response\_class

alias of Response

# aborter\_class

alias of Aborter

# jinja\_environment

alias of Environment

# app\_ctx\_globals\_class

alias of \_AppCtxGlobals

# config\_class

alias of Config

# testing

The testing flag. Set this to True to enable the test mode of Flask extensions (and in the future probably also Flask itself). For example this might activate test helpers that have an additional runtime cost which should not be enabled by default.

If this is enabled and PROPAGATE\_EXCEPTIONS is not changed from the default it's implicitly enabled.

This attribute can also be configured from the config with the TESTING configuration key. Defaults to False.

# secret\_key

If a secret key is set, cryptographic components can use this to sign cookies and other things. Set this to a complex random value when you want to use the secure cookie for instance.

This attribute can also be configured from the config with the SECRET\_KEY configuration key. Defaults to None.

#### property session\_cookie\_name: str

The name of the cookie set by the session interface.

Deprecated since version 2.2: Will be removed in Flask 2.3. Use app. config["SESSION\_COOKIE\_NAME"] instead.

#### permanent\_session\_lifetime

A timedelta which is used to set the expiration date of a permanent session. The default is 31 days which makes a permanent session survive for roughly one month.

This attribute can also be configured from the config with the PERMANENT\_SESSION\_LIFETIME configuration key. Defaults to timedelta(days=31)

# property send\_file\_max\_age\_default: timedelta | None

The default value for max\_age for send\_file(). The default is None, which tells the browser to use conditional requests instead of a timed cache.

Deprecated since version 2.2: Will be removed in Flask 2.3. Use app. config["SEND\_FILE\_MAX\_AGE\_DEFAULT"] instead.

Changed in version 2.0: Defaults to None instead of 12 hours.

# property use\_x\_sendfile: bool

Enable this to use the X-Sendfile feature, assuming the server supports it, from send\_file().

Deprecated since version 2.2: Will be removed in Flask 2.3. Use app.config["USE\_X\_SENDFILE"] instead.

# property json\_encoder: Type[JSONEncoder]

The JSON encoder class to use. Defaults to JSONEncoder.

Deprecated since version 2.2: Will be removed in Flask 2.3. Customize *json\_provider\_class* instead.

Added in version 0.10.

# property json\_decoder: Type[JSONDecoder]

The JSON decoder class to use. Defaults to JSONDecoder.

Deprecated since version 2.2: Will be removed in Flask 2.3. Customize <code>json\_provider\_class</code> instead.

Added in version 0.10.

# json\_provider\_class

alias of DefaultJSONProvider

# jinja\_options: dict = {}

Options that are passed to the Jinja environment in *create\_jinja\_environment()*. Changing these options after the environment is created (accessing *jinja\_env*) will have no effect.

Changed in version 1.1.0: This is a dict instead of an ImmutableDict to allow easier configuration.

```
default_config = {'APPLICATION_ROOT': '/', 'DEBUG': None, 'ENV': None,
   'EXPLAIN_TEMPLATE_LOADING': False, 'JSONIFY_MIMETYPE': None,
   'JSONIFY_PRETTYPRINT_REGULAR': None, 'JSON_AS_ASCII': None, 'JSON_SORT_KEYS': None,
   'MAX_CONTENT_LENGTH': None, 'MAX_COOKIE_SIZE': 4093, 'PERMANENT_SESSION_LIFETIME':
   datetime.timedelta(days=31), 'PREFERRED_URL_SCHEME': 'http', 'PROPAGATE_EXCEPTIONS':
   None, 'SECRET_KEY': None, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'SERVER_NAME': None,
   'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_HTTPONLY': True,
   'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_PATH': None,
   'SESSION_COOKIE_SAMESITE': None, 'SESSION_COOKIE_SECURE': False,
   'SESSION_REFRESH_EACH_REQUEST': True, 'TEMPLATES_AUTO_RELOAD': None, 'TESTING':
   False, 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS': False,
   'USE_X_SENDFILE': False}
```

Default configuration parameters.

#### url rule class

alias of Rule

#### url\_map\_class

alias of Map

# test\_client\_class: Type[FlaskClient] | None = None

The test\_client() method creates an instance of this test client class. Defaults to FlaskClient.

Added in version 0.7.

## test\_cli\_runner\_class: Type[FlaskCliRunner] | None = None

The CliRunner subclass, by default FlaskCliRunner that is used by test\_cli\_runner(). Its \_\_init\_\_ method should take a Flask app object as the first argument.

Added in version 1.0.

# session\_interface: SessionInterface = <flask.sessions.SecureCookieSessionInterface object>

the session interface to use. By default an instance of SecureCookieSessionInterface is used here.

Added in version 0.8.

```
__init__(import_name: str, static_url_path: str | None = None, static_folder: str | PathLike | None = 'static', static_host: str | None = None, host_matching: bool = False, subdomain_matching: bool = False, template_folder: str | PathLike | None = 'templates', instance_path: str | None = None, instance_relative_config: bool = False, root_path: str | None = None)
```

# instance\_path

Holds the path to the instance folder.

Added in version 0.8.

# config

The configuration dictionary as Config. This behaves exactly like a regular dictionary but supports additional methods to load a config from files.

#### aborter

An instance of *aborter\_class* created by *make\_aborter()*. This is called by **flask.abort()** to raise HTTP errors, and can be called directly as well.

Added in version 2.2: Moved from flask.abort, which calls this object.

#### json: JSONProvider

Provides access to JSON methods. Functions in flask.json will call methods on this provider when the application context is active. Used for handling JSON requests and responses.

An instance of *json\_provider\_class*. Can be customized by changing that attribute on a subclass, or by assigning to this attribute afterwards.

The default, DefaultJSONProvider, uses Python's built-in json library. A different provider can use a different JSON library.

Added in version 2.2.

# url\_build\_error\_handlers: t.List[t.Callable[[Exception, str, t.Dict[str, t.Any]], str]]

A list of functions that are called by <code>handle\_url\_build\_error()</code> when <code>url\_for()</code> raises a BuildError. Each function is called with error, endpoint and values. If a function returns None or raises a BuildError, it is skipped. Otherwise, its return value is returned by <code>url\_for</code>.

Added in version 0.9.

# before\_first\_request\_funcs: t.List[ft.BeforeFirstRequestCallable]

A list of functions that will be called at the beginning of the first request to this instance. To register a function, use the *before\_first\_request()* decorator.

Deprecated since version 2.2: Will be removed in Flask 2.3. Run setup code when creating the application instead.

Added in version 0.8.

# teardown\_appcontext\_funcs: t.List[ft.TeardownCallable]

A list of functions that are called when the application context is destroyed. Since the application context is also torn down if the request ends this is the place to store code that disconnects from databases.

Added in version 0.9.

# shell\_context\_processors: t.List[ft.ShellContextProcessorCallable]

A list of shell context processor functions that should be run when a shell context is created.

Added in version 0.11.

# blueprints: t.Dict[str, 'Blueprint']

Maps registered blueprint names to blueprint objects. The dict retains the order the blueprints were registered in. Blueprints can be registered multiple times, this dict does not track how often they were attached.

Added in version 0.7.

#### extensions: dict

a place where extensions can store application specific state. For example this is where an extension could store database engines and similar things.

The key must match the name of the extension module. For example in case of a "Flask-Foo" extension in flask\_foo, the key would be 'foo'.

Added in version 0.7.

#### url map

The Map for this instance. You can use this to change the routing converters after the class was created but before any routes are connected. Example:

#### property name: str

The name of the application. This is usually the import name with the difference that it's guessed from the run file if the import name is main. This name is used as a display name when Flask needs the name of the application. It can be set and overridden to change the value.

Added in version 0.8.

### property propagate\_exceptions: bool

Returns the value of the PROPAGATE\_EXCEPTIONS configuration value in case it's set, otherwise a sensible default is returned.

Deprecated since version 2.2: Will be removed in Flask 2.3.

Added in version 0.7.

# property logger: Logger

A standard Python Logger for the app, with the same name as *name*.

In debug mode, the logger's level will be set to DEBUG.

If there are no handlers configured, a default handler will be added. See /logging for more information.

Changed in version 1.1.0: The logger takes the same name as name rather than hard-coding "flask.app".

Changed in version 1.0.0: Behavior was simplified. The logger is always named "flask.app". The level is only set during configuration, it doesn't check app.debug each time. Only one format is used, not different ones depending on app.debug. No handlers are removed, and a handler is only added if no handlers are already configured.

Added in version 0.3.

#### property jinja\_env: Environment

The Jinja environment used to load templates.

The environment is created the first time this property is accessed. Changing *jinja\_options* after that will have no effect.

#### property got\_first\_request: bool

This attribute is set to True if the application started handling the first request.

Added in version 0.8.

```
make\_config(instance\_relative: bool = False) \rightarrow Config
```

Used to create the config attribute by the Flask constructor. The *instance\_relative* parameter is passed in from the constructor of Flask (there named *instance\_relative\_config*) and indicates if the config should be relative to the instance path or the root path of the application.

Added in version 0.8.

#### make\_aborter() → Aborter

Create the object to assign to *aborter*. That object is called by flask.abort() to raise HTTP errors, and can be called directly as well.

By default, this creates an instance of *aborter\_class*, which defaults to werkzeug.exceptions. Aborter.

Added in version 2.2.

#### auto\_find\_instance\_path() $\rightarrow$ str

Tries to locate the instance path if it was not provided to the constructor of the application class. It will basically calculate the path to a folder named instance next to your main file or the package.

Added in version 0.8.

#### open\_instance\_resource(resource: str, mode: str = 'rb') $\rightarrow$ IO

Opens a resource from the application's instance folder (*instance\_path*). Otherwise works like *open\_resource*(). Instance resources can also be opened for writing.

#### **Parameters**

- **resource** the name of the resource. To access resources within subfolders use forward slashes as separator.
- mode resource file opening mode, default is 'rb'.

# property templates\_auto\_reload: bool

Reload templates when they are changed. Used by *create\_jinja\_environment()*. It is enabled by default in debug mode.

Deprecated since version 2.2: Will be removed in Flask 2.3. Use app. config["TEMPLATES\_AUTO\_RELOAD"] instead.

Added in version 1.0: This property was added but the underlying config and behavior already existed.

# $create_{jinja\_environment}() \rightarrow Environment$

Create the Jinja environment based on <code>jinja\_options</code> and the various Jinja-related methods of the app. Changing <code>jinja\_options</code> after this will have no effect. Also adds Flask-related globals and filters to the environment.

Changed in version 0.11: Environment.auto\_reload set in accordance with TEMPLATES\_AUTO\_RELOAD configuration option.

Added in version 0.5.

# **create\_global\_jinja\_loader()** → DispatchingJinjaLoader

Creates the loader for the Jinja2 environment. Can be used to override just the loader and keeping the rest unchanged. It's discouraged to override this function. Instead one should override the <code>jinja\_loader()</code> function instead.

The global loader dispatches between the loaders of the application and the individual blueprints.

Added in version 0.7.

# $select_jinja_autoescape(filename: str) \rightarrow bool$

Returns True if autoescaping should be active for the given template name. If no template name is given, returns *True*.

Changed in version 2.2: Autoescaping is now enabled by default for .svg files.

Added in version 0.5.

#### $update\_template\_context(context: dict) \rightarrow None$

Update the template context with some commonly used variables. This injects request, session, config and g into the template context as well as everything template context processors want to inject. Note that the as of Flask 0.6, the original values in the context will not be overridden if a context processor decides to return a value with the same key.

#### **Parameters**

**context** – the context as a dictionary that is updated in place to add extra variables.

#### $make\_shell\_context() \rightarrow dict$

Returns the shell context for an interactive shell for this application. This runs all the registered shell context processors.

Added in version 0.11.

# property env: str

What environment the app is running in. This maps to the ENV config key.

# Do not enable development when deploying in production.

Default: 'production'

Deprecated since version 2.2: Will be removed in Flask 2.3.

#### property debug: bool

Whether debug mode is enabled. When using flask run to start the development server, an interactive debugger will be shown for unhandled exceptions, and the server will be reloaded when code changes. This maps to the DEBUG config key. It may not behave as expected if set late.

# Do not enable debug mode when deploying in production.

Default: False

```
run(host: str \mid None = None, port: int \mid None = None, debug: bool \mid None = None, load\_dotenv: bool = True, **options: Any) <math>\rightarrow None
```

Runs the application on a local development server.

Do not use run() in a production setting. It is not intended to meet security and performance requirements for a production server. Instead, see /deploying/index for WSGI server recommendations.

If the *debug* flag is set the server will automatically reload for code changes and show a debugger in case an exception happened.

If you want to run the application in debug mode, but disable the code execution on the interactive debugger, you can pass use\_evalex=False as parameter. This will keep the debugger's traceback screen active, but disable code execution.

It is not recommended to use this function for development with automatic reloading as this is badly supported. Instead you should be using the **flask** command line script's run support.

# **1** Keep in Mind

Flask will suppress any server error with a generic error page unless it is in debug mode. As such to enable just the interactive debugger without the code reloading, you have to invoke run() with debug=True and use\_reloader=False. Setting use\_debugger to True without being in debug mode won't catch any exceptions because there won't be any to catch.

#### **Parameters**

- host the hostname to listen on. Set this to '0.0.0.0' to have the server available externally as well. Defaults to '127.0.0.1' or the host in the SERVER\_NAME config variable if present.
- **port** the port of the webserver. Defaults to 5000 or the port defined in the SERVER\_NAME config variable if present.
- **debug** if given, enable or disable debug mode. See *debug*.
- load\_dotenv Load the nearest .env and .flaskenv files to set environment variables. Will also change the working directory to the directory containing the first file found.
- **options** the options to be forwarded to the underlying Werkzeug server. See werkzeug. serving.run\_simple() for more information.

Changed in version 1.0: If installed, python-dotenv will be used to load environment variables from .env and .flaskenv files.

The FLASK\_DEBUG environment variable will override debug.

Threaded mode is enabled by default.

Changed in version 0.10: The default port is now picked from the SERVER\_NAME variable.

```
test\_client(use\_cookies: bool = True, **kwargs: Any) \rightarrow FlaskClient
```

Creates a test client for this application. For information about unit testing head over to /testing.

Note that if you are testing for assertions or exceptions in your application code, you must set app.testing = True in order for the exceptions to propagate to the test client. Otherwise, the exception will be handled by the application (not visible to the test client) and the only indication of an AssertionError or other exception will be a 500 status code response to the test client. See the *testing* attribute. For example:

```
app.testing = True
client = app.test_client()
```

The test client can be used in a with block to defer the closing down of the context until the end of the with block. This is useful if you want to access the context locals for testing:

```
with app.test_client() as c:
    rv = c.get('/?vodka=42')
    assert request.args['vodka'] == '42'
```

Additionally, you may pass optional keyword arguments that will then be passed to the application's test\_client\_class constructor. For example:

```
from flask.testing import FlaskClient

class CustomClient(FlaskClient):
    def __init__(self, *args, **kwargs):
        self._authentication = kwargs.pop("authentication")
        super(CustomClient,self).__init__( *args, **kwargs)

app.test_client_class = CustomClient
    client = app.test_client(authentication='Basic ....')
```

See FlaskClient for more information.

Changed in version 0.4: added support for with block usage for the client.

Added in version 0.7: The *use\_cookies* parameter was added as well as the ability to override the client to be used by setting the *test\_client\_class* attribute.

Changed in version 0.11: Added \*\*kwargs to support passing additional keyword arguments to the constructor of test\_client\_class.

# **test\_cli\_runner**(\*\*kwargs: Any) → FlaskCliRunner

Create a CLI runner for testing CLI commands. See testing-cli.

Returns an instance of test\_cli\_runner\_class, by default FlaskCliRunner. The Flask app object is passed as the first argument.

Added in version 1.0.

#### **register\_blueprint**(*blueprint*: *Blueprint*, \*\*options: Any) $\rightarrow$ None

Register a Blueprint on the application. Keyword arguments passed to this method will override the defaults set on the blueprint.

Calls the blueprint's register() method after recording the blueprint in the application's *blueprints*.

#### **Parameters**

- **blueprint** The blueprint to register.
- **url\_prefix** Blueprint routes will be prefixed with this.
- **subdomain** Blueprint routes will match on this subdomain.
- **url\_defaults** Blueprint routes will use these default values for view arguments.
- **options** Additional keyword arguments are passed to BlueprintSetupState. They can be accessed in record() callbacks.

Changed in version 2.0.1: The name option can be used to change the (pre-dotted) name the blueprint is registered with. This allows the same blueprint to be registered multiple times with unique names for url\_for.

Added in version 0.7.

# $iter\_blueprints() \rightarrow ValuesView[Blueprint]$

Iterates over all blueprints by the order they were registered.

Added in version 0.11.

add\_url\_rule(rule: str, endpoint: str | None = None, view\_func: Callable[[...], Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | WSGIApplication] | Callable[[...], Awaitable[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | WSGIApplication]] | None = None,  $provide\_automatic\_options: bool \mid None = None, **options: Any) \rightarrow None$ 

Register a rule for routing incoming requests and building URLs. The *route()* decorator is a shortcut to call this with the view\_func argument. These are equivalent:

```
@app.route("/")
def index():
    ...
```

```
def index():
    ...
app.add_url_rule("/", view_func=index)
```

See url-route-registrations.

The endpoint name for the route defaults to the name of the view function if the endpoint parameter isn't passed. An error will be raised if a function has already been registered for the endpoint.

The methods parameter defaults to ["GET"]. HEAD is always added automatically, and OPTIONS is added automatically by default.

view\_func does not necessarily need to be passed, but if the rule should participate in routing an endpoint name must be associated with a view function at some point with the <code>endpoint()</code> decorator.

```
app.add_url_rule("/", endpoint="index")
@app.endpoint("index")
def index():
    ...
```

If view\_func has a required\_methods attribute, those methods are added to the passed and automatic methods. If it has a provide\_automatic\_methods attribute, it is used as the default if the parameter is not passed.

#### **Parameters**

- **rule** The URL rule string.
- **endpoint** The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view\_func.\_\_name\_\_.
- **view\_func** The view function to associate with the endpoint name.
- **provide\_automatic\_options** Add the OPTIONS method and respond to OPTIONS requests automatically.
- **options** Extra options passed to the Rule object.

```
template_filter(name: str | None = None) → Callable[[T_template_filter], T_template_filter]
```

A decorator that is used to register custom template filter. You can specify a name for the filter, otherwise the function name will be used. Example:

```
@app.template_filter()
def reverse(s):
   return s[::-1]
```

# **Parameters**

**name** – the optional name of the filter, otherwise the function name will be used.

```
add\_template\_filter(f: Callable[[...], Any], name: str | None = None) \rightarrow None
```

Register a custom template filter. Works exactly like the template\_filter() decorator.

#### **Parameters**

**name** – the optional name of the filter, otherwise the function name will be used.

```
template\_test(name: str \mid None = None) \rightarrow Callable[[T\_template\_test], T\_template\_test]
```

A decorator that is used to register custom template test. You can specify a name for the test, otherwise the function name will be used. Example:

```
@app.template_test()
def is_prime(n):
    if n == 2:
        return True
    for i in range(2, int(math.ceil(math.sqrt(n))) + 1):
        if n % i == 0:
            return False
    return True
```

Added in version 0.10.

#### **Parameters**

**name** – the optional name of the test, otherwise the function name will be used.

```
add_template_test(f: Callable[[...], bool], name: str | None = None) \rightarrow None
```

Register a custom template test. Works exactly like the template\_test() decorator.

Added in version 0.10.

### **Parameters**

**name** – the optional name of the test, otherwise the function name will be used.

```
template_global(name: str | None = None) \rightarrow Callable[[T_template_global], T_template_global]
```

A decorator that is used to register a custom template global function. You can specify a name for the global function, otherwise the function name will be used. Example:

```
@app.template_global()
def double(n):
   return 2 * n
```

Added in version 0.10.

#### **Parameters**

**name** – the optional name of the global function, otherwise the function name will be used.

```
add_template_global(f: Callable[[...], Any], name: str | None = None) \rightarrow None
```

Register a custom template global function. Works exactly like the template\_global() decorator.

Added in version 0.10.

#### **Parameters**

**name** – the optional name of the global function, otherwise the function name will be used.

# $before\_first\_request(f: T\_before\_first\_request) \rightarrow T\_before\_first\_request$

Registers a function to be run before the first request to this instance of the application.

The function will be called without any arguments and its return value is ignored.

Deprecated since version 2.2: Will be removed in Flask 2.3. Run setup code when creating the application instead.

Added in version 0.8.

#### $teardown\_appcontext(f: T\_teardown) \rightarrow T\_teardown$

Registers a function to be called when the application context is popped. The application context is typically popped after the request context for each request, at the end of CLI commands, or after a manually pushed context ends.

```
with app.app_context():
...
```

When the with block exits (or ctx.pop() is called), the teardown functions are called just before the app context is made inactive. Since a request context typically also manages an application context it would also be called when you pop a request context.

When a teardown function was called because of an unhandled exception it will be passed an error object. If an *errorhandler()* is registered, it will handle the exception and the teardown will not receive it.

Teardown functions must avoid raising exceptions. If they execute code that might fail they must surround that code with a try/except block and log any errors.

The return values of teardown functions are ignored.

Added in version 0.9.

```
\textbf{shell\_context\_processor}(\textit{f: T\_shell\_context\_processor}) \rightarrow \textbf{T\_shell\_context\_processor})
```

Registers a shell context processor function.

Added in version 0.11.

```
handle_http_exception(e: HTTPException) → HTTPException | Response | str | bytes | List[Any] |
```

Mapping[str, Any] | Iterator[str] | Iterator[bytes] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str, ...]]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | WSGIApplication

Handles an HTTP exception. By default this will invoke the registered error handlers and fall back to returning the exception as response.

Changed in version 1.0.3: RoutingException, used internally for actions such as slash redirects during routing, is not passed to error handlers.

Changed in version 1.0: Exceptions are looked up by code *and* by MRO, so HTTPException subclasses can be handled with a catch-all handler for the base HTTPException.

Added in version 0.3.

# $trap_http_exception(e: Exception) \rightarrow bool$

Checks if an HTTP exception should be trapped or not. By default this will return False for all exceptions except for a bad request key error if TRAP\_BAD\_REQUEST\_ERRORS is set to True. It also returns True if TRAP\_HTTP\_EXCEPTIONS is set to True.

This is called for all HTTP exceptions raised by a view function. If it returns True for any exception the error handler for this exception is not called and it shows up as regular exception in the traceback. This is helpful for debugging implicitly raised HTTP exceptions.

Changed in version 1.0: Bad request errors are not trapped by default in debug mode.

Added in version 0.8.

handle\_user\_exception(e: Exception) → HTTPException | Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[str] | Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str, ...]]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | WSGIApplication

This method is called whenever an exception occurs that should be handled. A special case is HTTPException which is forwarded to the <code>handle\_http\_exception()</code> method. This function will either return a response value or reraise the exception with the same traceback.

Changed in version 1.0: Key errors raised from request data like form show the bad key in debug mode rather than a generic bad request message.

Added in version 0.7.

#### **handle\_exception**(e: Exception) $\rightarrow$ Response

Handle an exception that did not have an error handler associated with it, or that was raised from an error handler. This always causes a 500 InternalServerError.

Always sends the got\_request\_exception signal.

If *propagate\_exceptions* is True, such as in debug mode, the error will be re-raised so that the debugger can display it. Otherwise, the original exception is logged, and an InternalServerError is returned.

If an error handler is registered for InternalServerError or 500, it will be used. For consistency, the handler will always receive the InternalServerError. The original unhandled exception is available as e.original\_exception.

Changed in version 1.1.0: Always passes the InternalServerError instance to the handler, setting original\_exception to the unhandled error.

Changed in version 1.1.0: after\_request functions and other finalization is done even for the default 500 response when there is no handler.

Added in version 0.3.

 $log\_exception(exc\_info: Tuple[type, BaseException, TracebackType] | Tuple[None, None, None]) \rightarrow None Logs an exception. This is called by <math>handle\_exception()$  if debugging is disabled and right before the handler is called. The default implementation logs the exception as error on the logger.

Added in version 0.8.

dispatch\_request() → Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes] |

Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes],

Headers | Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] |

Tuple[str, ...]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] |

Iterator[str] | Iterator[bytes], int] | Tuple[Response | str | bytes | List[Any] | Mapping[str,

Any] | Iterator[str] | Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str,

...]] | Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | WSGIApplication

Does the request dispatching. Matches the URL and returns the return value of the view or error handler. This does not have to be a response object. In order to convert the return value to a proper response object, call <code>make\_response()</code>.

Changed in version 0.7: This no longer does the exception handling, this code was moved to the new full\_dispatch\_request().

# $full\_dispatch\_request() \rightarrow Response$

Dispatches the request and on top of that performs request pre and postprocessing as well as HTTP exception catching and error handling.

Added in version 0.7.

Given the return value from a view function this finalizes the request by converting it into a response and invoking the postprocessing functions. This is invoked for both normal request dispatching as well as error handlers.

Because this means that it might be called as a result of a failure a special safe mode is available which can be enabled with the *from\_error\_handler* flag. If enabled, failures in response processing will be logged and otherwise ignored.

#### **Internal**

# $make\_default\_options\_response() \rightarrow Response$

This method is called to create the default OPTIONS response. This can be changed through subclassing to change the default behavior of OPTIONS responses.

Added in version 0.7.

# **should\_ignore\_error**( $error: BaseException \mid None$ ) $\rightarrow$ bool

This is called to figure out if an error should be ignored or not as far as the teardown system is concerned. If this function returns True then the teardown handlers will not be passed the error.

Added in version 0.10.

```
ensure_sync(func: Callable) → Callable
```

Ensure that the function is synchronous for WSGI workers. Plain def functions are returned as-is. async def functions are wrapped to run and wait for the response.

Override this method to change how the app runs async views.

Added in version 2.0.

```
async_to_sync(func: Callable[[...], Coroutine]) → Callable[[...], Any]
```

Return a sync function that will run the coroutine function.

```
result = app.async_to_sync(func)(*args, **kwargs)
```

Override this method to change how the app converts async code to be synchronously callable.

Added in version 2.0.

```
url_for(endpoint: str, *, _anchor: str | None = None, _method: str | None = None, _scheme: str | None = None, _external: bool | None = None, **values: Any) → str
```

Generate a URL to the given endpoint with the given values.

This is called by flask.url\_for(), and can be called directly as well.

An *endpoint* is the name of a URL rule, usually added with <code>@app.route()</code>, and usually the same name as the view function. A route defined in a <code>Blueprint</code> will prepend the blueprint's name separated by a . to the endpoint.

In some cases, such as email messages, you want URLs to include the scheme and domain, like https://example.com/hello. When not in an active request, URLs will be external by default, but this requires setting SERVER\_NAME so Flask knows what domain to use. APPLICATION\_ROOT and PREFERRED\_URL\_SCHEME should also be configured as needed. This config is only used when not in an active request.

Functions can be decorated with url\_defaults() to modify keyword arguments before the URL is built.

If building fails for some reason, such as an unknown endpoint or incorrect values, the app's handle\_url\_build\_error() method is called. If that returns a string, that is returned, otherwise a BuildError is raised.

#### **Parameters**

- **endpoint** The endpoint name associated with the URL to generate. If this starts with a ., the current blueprint name (if any) will be used.
- **\_anchor** If given, append this as #anchor to the URL.
- \_method If given, generate the URL associated with this method for the endpoint.
- **\_scheme** If given, the URL will have this scheme if it is external.
- **\_external** If given, prefer the URL to be internal (False) or require it to be external (True). External URLs include the scheme and domain. When not in an active request, URLs are external by default.
- **values** Values to use for the variable parts of the URL rule. Unknown keys are appended as query string arguments, like ?a=b&c=d.

Added in version 2.2: Moved from flask.url\_for, which calls this method.

**redirect**(*location:* str, code: int = 302)  $\rightarrow$  Response

Create a redirect response object.

This is called by flask.redirect(), and can be called directly as well.

# **Parameters**

- location The URL to redirect to.
- **code** The status code for the redirect.

Added in version 2.2: Moved from flask.redirect, which calls this method.

make\_response( $rv: Response \mid str \mid bytes \mid List[Any] \mid Mapping[str, Any] \mid Iterator[str] \mid Iterator[bytes] \mid$   $Tuple[Response \mid str \mid bytes \mid List[Any] \mid Mapping[str, Any] \mid Iterator[str] \mid Iterator[bytes],$   $Headers \mid Mapping[str, str \mid List[str] \mid Tuple[str, ...]] \mid Sequence[Tuple[str, str \mid List[str] \mid$   $Tuple[str, ...]]] \mid Tuple[Response \mid str \mid bytes \mid List[Any] \mid Mapping[str, Any] \mid Iterator[str] \mid$   $Iterator[bytes], int] \mid Tuple[Response \mid str \mid bytes \mid List[Any] \mid Mapping[str, Any] \mid$   $Iterator[str] \mid Iterator[bytes], int, Headers \mid Mapping[str, str \mid List[str] \mid Tuple[str, ...]] \mid$   $Sequence[Tuple[str, str \mid List[str] \mid Tuple[str, ...]]] \mid WSGIApplication) \rightarrow Response$ 

Convert the return value from a view function to an instance of *response\_class*.

## **Parameters**

rv - the return value from the view function. The view function must return a response.
Returning None, or the view ending without returning, is not allowed. The following types are allowed for view\_rv:

#### str

A response object is created with the string encoded to UTF-8 as the body.

#### bytes

A response object is created with the bytes as the body.

#### dict

A dictionary that will be jsonify'd before being returned.

#### list

A list that will be jsonify'd before being returned.

## generator or iterator

A generator that returns str or bytes to be streamed as the response.

#### tuple

Either (body, status, headers), (body, status), or (body, headers), where body is any of the other types allowed here, status is a string or an integer, and headers is a dictionary or a list of (key, value) tuples. If body is a *response\_class* instance, status overwrites the exiting value and headers are extended.

## response\_class

The object is returned unchanged.

# other Response class

The object is coerced to response\_class.

#### callable()

The function is called as a WSGI application. The result is used to create a response object.

Changed in version 2.2: A generator will be converted to a streaming response. A list will be converted to a JSON response.

Changed in version 1.1: A dict will be converted to a JSON response.

Changed in version 0.9: Previously a tuple was interpreted as the arguments for the response object.

# **create\_url\_adapter**(request: Request | None) → MapAdapter | None

Creates a URL adapter for the given request. The URL adapter is created at a point where the request context is not yet set up so the request is passed explicitly.

Added in version 0.6.

Changed in version 0.9: This can now also be called without a request object when the URL adapter is created for the application context.

Changed in version 1.0: SERVER\_NAME no longer implicitly enables subdomain matching. Use subdomain\_matching instead.

# **after\_request**( $f: T\_after\_request$ ) $\rightarrow$ T\_after\_request

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after\_request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown\_request() for that.

This is available on both app and blueprint objects. When used on an app, this executes after every request. When used on a blueprint, this executes after every request that the blueprint handles. To register with a blueprint and execute after every request, use Blueprint.after\_app\_request().

```
before_request(f: T\_before\_request) \rightarrow T_before_request
```

Register a function to run before each request.

For example, this can be used to open a database connection, or to load the logged in user from the session.

```
@app.before_request
def load_user():
   if "user_id" in session:
        g.user = db.session.get(session["user_id"])
```

The function will be called without any arguments. If it returns a non-None value, the value is handled as if it was the return value from the view, and further request handling is stopped.

This is available on both app and blueprint objects. When used on an app, this executes before every request. When used on a blueprint, this executes before every request that the blueprint handles. To register with a blueprint and execute before every request, use Blueprint.before\_app\_request().

```
context\_processor(f: T\_template\_context\_processor) \rightarrow T\_template\_context\_processor
```

Registers a template context processor function. These functions run before rendering a template. The keys of the returned dict are added as variables available in the template.

This is available on both app and blueprint objects. When used on an app, this is called for every rendered template. When used on a blueprint, this is called for templates rendered from the blueprint's views. To register with a blueprint and affect every template, use Blueprint.app\_context\_processor().

```
delete(rule: str, **options: Any) \rightarrow Callable[[T_route], T_route]
Shortcut for route() with methods=["DELETE"].
```

Added in version 2.0.

```
endpoint(endpoint: str) \rightarrow Callable[[F], F]
```

Decorate a view function to register it for the given endpoint. Used if a rule is added without a view\_func with add\_url\_rule().

```
app.add_url_rule("/ex", endpoint="example")
@app.endpoint("example")
def example():
...
```

### **Parameters**

 $\label{eq:continuous} \textbf{endpoint} - The \ endpoint \ name \ to \ associate \ with \ the \ view \ function.$ 

```
errorhandler(code\_or\_exception: Type[Exception] \mid int) \rightarrow Callable[[T\_error\_handler], T\_error\_handler] Register a function to handle errors by code or exception class.
```

A decorator that is used to register a function given an error code. Example:

```
@app.errorhandler(404)
def page_not_found(error):
    return 'This page does not exist', 404
```

You can also register handlers for arbitrary exceptions:

```
@app.errorhandler(DatabaseError)
def special_exception_handler(error):
    return 'Database connection failed', 500
```

This is available on both app and blueprint objects. When used on an app, this can handle errors from every request. When used on a blueprint, this can handle errors from requests that the blueprint handles. To register with a blueprint and affect every request, use Blueprint.app\_errorhandler().

Added in version 0.7: Use register\_error\_handler() instead of modifying error\_handler\_spec directly, for application wide error handlers.

Added in version 0.7: One can now additionally also register custom exception types that do not necessarily have to be a subclass of the HTTPException class.

# **Parameters**

**code\_or\_exception** – the code as integer for the handler, or an arbitrary exception

```
get(rule: str, **options: Any) \rightarrow Callable[[T_route], T_route]
Shortcut for route() with methods=["GET"].
```

Added in version 2.0.

```
get\_send\_file\_max\_age(filename: str \mid None) \rightarrow int \mid None
```

Used by send\_file() to determine the max\_age cache value for a given file path if it wasn't passed.

By default, this returns SEND\_FILE\_MAX\_AGE\_DEFAULT from the configuration of current\_app. This defaults to None, which tells the browser to use conditional requests instead of a timed cache, which is usually preferable.

Changed in version 2.0: The default configuration is None instead of 12 hours.

Added in version 0.9.

```
property has_static_folder: bool
```

True if static\_folder is set.

Added in version 0.5.

```
inject\_url\_defaults(endpoint: str, values: dict) \rightarrow None
```

Injects the URL defaults for the given endpoint directly into the values dictionary passed. This is used internally and automatically called on URL building.

Added in version 0.7.

#### property jinja\_loader: FileSystemLoader | None

The Jinja loader for this object's templates. By default this is a class jinja2.loaders. FileSystemLoader to template\_folder if it is set.

Added in version 0.5.

```
open_resource(resource: str, mode: str = 'rb') \rightarrow IO
```

Open a resource file relative to *root\_path* for reading.

For example, if the file schema.sql is next to the file app.py where the Flask app is defined, it can be opened with:

```
with app.open_resource("schema.sql") as f:
    conn.executescript(f.read())
```

#### **Parameters**

- **resource** Path to the resource relative to **root\_path**.
- **mode** Open the file in this mode. Only reading is supported, valid values are "r" (or "rt") and "rb".

register\_error\_handler(code\_or\_exception: Type[Exception] | int, f: Callable[[Any], Response | str |
bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes] |
Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] |
Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] |
Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[str] |
Iterator[bytes], int, Headers | Mapping[str, str | List[str] | Tuple[str, ...]] |
Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]]] | WSGIApplication]) →
None

Alternative error attach function to the *errorhandler()* decorator that is more straightforward to use for non decorator usage.

Added in version 0.7.

```
route(rule: str, **options: Any) \rightarrow Callable[[T route], T route]
```

Decorate a view function to register it with the given URL rule and options. Calls add\_url\_rule(), which has more details about the implementation.

```
@app.route("/")
def index():
    return "Hello, World!"
```

See url-route-registrations.

The endpoint name for the route defaults to the name of the view function if the endpoint parameter isn't passed.

The methods parameter defaults to ["GET"]. HEAD and OPTIONS are added automatically.

# **Parameters**

- **rule** The URL rule string.
- options Extra options passed to the Rule object.

```
send_static_file(filename: str) \rightarrow Response
```

The view function used to serve files from  $static\_folder$ . A route is automatically registered for this view at  $static\_url\_path$  if  $static\_folder$  is set.

Added in version 0.5.

```
property static_folder: str | None
```

The absolute path to the configured static folder. None if no static folder is set.

#### property static\_url\_path: str | None

The URL prefix that the static route will be accessible from.

If it was not configured during init, it is derived from static\_folder.

```
teardown_request(f: T \ teardown) \rightarrow T teardown
```

Register a function to be called when the request context is popped. Typically this happens at the end of each request, but contexts may be pushed manually as well during testing.

```
with app.test_request_context():
...
```

When the with block exits (or ctx.pop() is called), the teardown functions are called just before the request context is made inactive.

When a teardown function was called because of an unhandled exception it will be passed an error object. If an *errorhandler()* is registered, it will handle the exception and the teardown will not receive it.

Teardown functions must avoid raising exceptions. If they execute code that might fail they must surround that code with a try/except block and log any errors.

The return values of teardown functions are ignored.

This is available on both app and blueprint objects. When used on an app, this executes after every request. When used on a blueprint, this executes after every request that the blueprint handles. To register with a blueprint and execute after every request, use Blueprint.teardown\_app\_request().

```
url\_defaults(f: T\_url\_defaults) \rightarrow T\_url\_defaults
```

Callback function for URL defaults for all view functions of the application. It's called with the endpoint and values and should update the values passed in place.

This is available on both app and blueprint objects. When used on an app, this is called for every request. When used on a blueprint, this is called for requests that the blueprint handles. To register with a blueprint and affect every request, use Blueprint.app\_url\_defaults().

```
url\_value\_preprocessor(f: T\_url\_value\_preprocessor) \rightarrow T\_url\_value\_preprocessor
```

Register a URL value preprocessor function for all view functions in the application. These functions will be called before the *before\_request()* functions.

The function can modify the values captured from the matched url before they are passed to the view. For example, this can be used to pop a common language code value and place it in g rather than pass it to every view.

The function is passed the endpoint name and values dict. The return value is ignored.

This is available on both app and blueprint objects. When used on an app, this is called for every request. When used on a blueprint, this is called for requests that the blueprint handles. To register with a blueprint and affect every request, use Blueprint.app\_url\_value\_preprocessor().

## import\_name

The name of the package or module that this object belongs to. Do not change this once it is set by the constructor.

#### template\_folder

The path to the templates folder, relative to *root\_path*, to add to the template loader. None if templates should not be added.

# root\_path

Absolute path to the package on the filesystem. Used to look up resources contained in the package.

# cli

The Click command group for registering CLI commands for this object. The commands are available from the flask command once the application has been discovered and blueprints have been registered.

## view\_functions: t.Dict[str, t.Callable]

A dictionary mapping endpoint names to view functions.

To register a view function, use the *route()* decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# error\_handler\_spec: t.Dict[ft.AppOrBlueprintKey, t.Dict[t.Optional[int], t.Dict[t.Type[Exception], ft.ErrorHandlerCallable]]]

A data structure of registered error handlers, in the format {scope: {code: {class: handler}}}. The scope key is the name of a blueprint the handlers are active for, or None for all requests. The code key is the HTTP status code for HTTPException, or None for other exceptions. The innermost dictionary maps exception classes to handler functions.

To register an error handler, use the *errorhandler()* decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# before\_request\_funcs: t.Dict[ft.AppOrBlueprintKey, t.List[ft.BeforeRequestCallable]]

A data structure of functions to call at the beginning of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the *before\_request()* decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# after\_request\_funcs: t.Dict[ft.App0rBlueprintKey, t.List[ft.AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after\_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

## teardown\_request\_funcs: t.Dict[ft.AppOrBlueprintKey, t.List[ft.TeardownCallable]]

A data structure of functions to call at the end of each request even if an exception is raised, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the teardown\_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# template\_context\_processors: t.Dict[ft.AppOrBlueprintKey, t.List[ft.TemplateContextProcessorCallable]]

A data structure of functions to call to pass extra context values when rendering templates, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the *context\_processor()* decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# url\_value\_preprocessors: t.Dict[ft.AppOrBlueprintKey, t.List[ft.URLValuePreprocessorCallable]]

A data structure of functions to call to modify the keyword arguments passed to the view function, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the *url\_value\_preprocessor()* decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# url\_default\_functions: t.Dict[ft.AppOrBlueprintKey, t.List[ft.URLDefaultCallable]]

A data structure of functions to call to modify the keyword arguments when generating URLs, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the url\_defaults() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

# **handle\_url\_build\_error**(error: BuildError, endpoint: str, values: Dict[str, Any]) $\rightarrow$ str

Called by  $url\_for()$  if a BuildError was raised. If this returns a value, it will be returned by  $url\_for$ , otherwise the error will be re-raised.

Each function in *url\_build\_error\_handlers* is called with error, endpoint and values. If a function returns None or raises a BuildError, it is skipped. Otherwise, its return value is returned by *url\_for*.

#### **Parameters**

- **error** The active BuildError being handled.
- **endpoint** The endpoint being built.
- **values** The keyword arguments passed to url\_for.

```
preprocess_request() → Response | str | bytes | List[Any] | Mapping[str, Any] | Iterator[str] |

Iterator[bytes] | Tuple[Response | str | bytes | List[Any] | Mapping[str, Any] |

Iterator[str] | Iterator[bytes], Headers | Mapping[str, str | List[str] | Tuple[str, ...]] |

Sequence[Tuple[str, str | List[str] | Tuple[str, ...]]] | Tuple[Response | str | bytes |

List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int] | Tuple[Response | str |

bytes | List[Any] | Mapping[str, Any] | Iterator[str] | Iterator[bytes], int, Headers |

Mapping[str, str | List[str] | Tuple[str, ...]] | Sequence[Tuple[str, str | List[str] |

Tuple[str, ...]]] | WSGIApplication | None
```

Called before the request is dispatched. Calls *url\_value\_preprocessors* registered with the app and the current blueprint (if any). Then calls *before\_request\_funcs* registered with the app and the blueprint.

If any before\_request() handler returns a non-None value, the value is handled as if it was the return value from the view, and further request handling is stopped.

# $process\_response(response: Response) \rightarrow Response$

Can be overridden in order to modify the response object before it's sent to the WSGI server. By default this will call all the *after\_request()* decorated functions.

Changed in version 0.5: As of Flask 0.5 the functions registered for after request execution are called in reverse order of registration.

#### **Parameters**

response – a response\_class object.

#### Returns

a new response object or the same, has to be an instance of response\_class.

#### $do_{teardown\_request(exc: BaseException | None = < object object>) \rightarrow None$

Called after the request is dispatched and the response is returned, right before the request context is popped.

This calls all functions decorated with teardown\_request(), and Blueprint.teardown\_request() if a blueprint handled the request. Finally, the request\_tearing\_down signal is sent.

This is called by RequestContext.pop(), which may be delayed during testing to maintain access to resources.

#### **Parameters**

**exc** – An unhandled exception raised while dispatching the request. Detected from the current exception information if not passed. Passed to each teardown function.

Changed in version 0.9: Added the exc argument.

# $do_teardown_appcontext(exc: BaseException | None = < object object>) \rightarrow None$

Called right before the application context is popped.

When handling a request, the application context is popped after the request context. See do\_teardown\_request().

This calls all functions decorated with teardown\_appcontext(). Then the appcontext\_tearing\_down signal is sent.

This is called by AppContext.pop().

Added in version 0.9.

# $app\_context() \rightarrow AppContext$

Create an AppContext. Use as a with block to push the context, which will make current\_app point at this application.

An application context is automatically pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

```
with app.app_context():
   init_db()
```

See /appcontext.

Added in version 0.9.

# $request\_context(environ: dict) \rightarrow RequestContext$

Create a RequestContext representing a WSGI environment. Use a with block to push the context, which will make request point at this request.

See /regcontext.

Typically you should not call this from your own code. A request context is automatically pushed by the <code>wsgi\_app()</code> when handling a request. Use <code>test\_request\_context()</code> to create an environment and context instead of this method.

# **Parameters**

environ - a WSGI environment

# $test\_request\_context(*args: Any, **kwargs: Any) \rightarrow RequestContext$

Create a RequestContext for a WSGI environment created from the given values. This is mostly useful during testing, where you may want to run a function that uses request data without dispatching a full request.

See /reqcontext.

Use a with block to push the context, which will make request point at the request for the created environment.

```
with test_request_context(...):
    generate_report()
```

When using the shell, it may be easier to push and pop the context manually to avoid indentation.

```
ctx = app.test_request_context(...)
ctx.push()
...
ctx.pop()
```

Takes the same arguments as Werkzeug's EnvironBuilder, with some defaults from the application. See the linked Werkzeug docs for most of the available arguments. Flask-specific behavior is listed here.

#### **Parameters**

- path URL path being requested.
- base\_url Base URL where the app is being served, which path is relative to. If not given, built from PREFERRED\_URL\_SCHEME, subdomain, SERVER\_NAME, and APPLICATION\_ROOT.
- **subdomain** Subdomain name to append to SERVER\_NAME.
- url\_scheme Scheme to use instead of PREFERRED\_URL\_SCHEME.
- data The request body, either as a string or a dict of form keys and values.
- **json** If given, this is serialized as JSON and passed as data. Also defaults content\_type to application/json.
- **args** other positional arguments passed to EnvironBuilder.
- **kwargs** other keyword arguments passed to EnvironBuilder.

```
wsgi_app(environ: dict, start_response: Callable) \rightarrow Any
```

The actual WSGI application. This is not implemented in  $\__call\_\_()$  so that middlewares can be applied without losing a reference to the app object. Instead of doing this:

```
app = MyMiddleware(app)
```

It's a better idea to do this instead:

```
app.wsgi_app = MyMiddleware(app.wsgi_app)
```

Then you still have the original application object around and can continue to call methods on it.

Changed in version 0.7: Teardown events for the request and app contexts are called even if an unhandled error occurs. Other events may not be called depending on when an error occurs during dispatch. See callbacks-and-errors.

#### **Parameters**

- **environ** A WSGI environment.
- **start\_response** A callable accepting a status code, a list of headers, and an optional exception context to start the response.

```
__call__(environ: dict, start_response: Callable) \rightarrow Any
```

The WSGI server calls the Flask application object as the WSGI application. This calls wsgi\_app(), which can be wrapped to apply middleware.

# AFL.automation.APIServer.APIServer.IPVersion

```
Bases: Enum
__init__(*args, **kwds)
```

# **Attributes**

```
V40nly
V60nly
All
```

```
V40nly = 1
```

V60nly = 2

A11 = 3

```
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching *name*.

```
classmethod __iter__()
```

Return members in definition order.

# classmethod \_\_len\_\_()

Return the number of members (no aliases)

# AFL.automation.APIServer.APIServer.JWTManager

Bases: object

An object used to hold JWT settings and callback functions for the Flask-JWT-Extended extension.

Instances of JWTManager are not bound to specific apps, so you can create one in the main body of your code and then bind it to your app in a factory function.

```
__init__(app: Flask \mid None = None, add_context_processor: <math>bool = False) \rightarrow None
```

Create the JWTManager instance. You can either pass a flask application in directly here to register this extension with the flask app, or call init\_app after creating this object (in a factory pattern).

#### **Parameters**

- app The Flask Application object
- **add\_context\_processor** Controls if *current\_user* is should be added to flasks template context (and thus be available for use in Jinja templates). Defaults to False.

# **Methods**

<pre>init([app, add_context_processor])</pre>	Create the JWTManager instance.
additional_claims_loader(callback)	This decorator sets the callback function used to add additional claims when creating a JWT.
additional_headers_loader(callback)	This decorator sets the callback function used to add
	additional headers when creating a JWT.
decode_key_loader(callback)	This decorator sets the callback function for dynamically setting the JWT decode key based on the <b>UN-VERIFIED</b> contents of the token.
encode_key_loader(callback)	This decorator sets the callback function for dynamically setting the JWT encode key based on the tokens identity.
<pre>expired_token_loader(callback)</pre>	This decorator sets the callback function for returning
	a custom response when an expired JWT is encountered.
<pre>init_app(app[, add_context_processor])</pre>	Register this extension with the flask app.
<pre>invalid_token_loader(callback)</pre>	This decorator sets the callback function for returning a custom response when an invalid JWT is encountered.
<pre>needs_fresh_token_loader(callback)</pre>	This decorator sets the callback function for returning
	a custom response when a valid and non-fresh token
	is used on an endpoint that is marked as fresh=True.
revoked_token_loader(callback)	This decorator sets the callback function for returning a custom response when a revoked token is encoun- tered.
token_in_blocklist_loader(callback)	This decorator sets the callback function used to check if a JWT has been revoked.
<pre>token_verification_failed_loader(callback)</pre>	This decorator sets the callback function used to re-
	turn a custom response when the claims verification check fails.
token_verification_loader(callback)	This decorator sets the callback function used for custom verification of a valid JWT.
unauthorized_loader(callback)	This decorator sets the callback function used to return a custom response when no JWT is present.
user_identity_loader(callback)	This decorator sets the callback function used to convert an identity to a string when creating JWTs.
user_lookup_error_loader(callback)	This decorator sets the callback function used to return a custom response when loading a user via
	user_lookup_loader() fails.
user_lookup_loader(callback)	This decorator sets the callback function used to convert a JWT into a python object that can be used in a protected endpoint.
	ргоссова спаропи.

 $\_$ \_init $\_$ (app: Flask | None = None, add $\_$ context $\_$ processor: bool = False)  $\rightarrow$  None

Create the JWTManager instance. You can either pass a flask application in directly here to register this extension with the flask app, or call init\_app after creating this object (in a factory pattern).

## **Parameters**

- app The Flask Application object
- **add\_context\_processor** Controls if *current\_user* is should be added to flasks template context (and thus be available for use in Jinja templates). Defaults to False.

#### $init_app(app: Flask, add\_context\_processor: bool = False) \rightarrow None$

Register this extension with the flask app.

#### **Parameters**

- app The Flask Application object
- **add\_context\_processor** Controls if *current\_user* is should be added to flasks template context (and thus be available for use in Jinja templates). Defaults to False.

#### **additional\_claims\_loader**(*callback: Callable*) → Callable

This decorator sets the callback function used to add additional claims when creating a JWT. The claims returned by this function will be merged with any claims passed in via the additional\_claims argument to create\_access\_token() or create\_refresh\_token().

The decorated function must take one argument.

The argument is the identity that was used when creating a JWT.

The decorated function must return a dictionary of claims to add to the JWT.

## **additional\_headers\_loader**(*callback: Callable*) → Callable

This decorator sets the callback function used to add additional headers when creating a JWT. The headers returned by this function will be merged with any headers passed in via the additional\_headers argument to create\_access\_token() or create\_refresh\_token().

The decorated function must take one argument.

The argument is the identity that was used when creating a JWT.

The decorated function must return a dictionary of headers to add to the JWT.

## **decode\_key\_loader**(*callback: Callable*) → Callable

This decorator sets the callback function for dynamically setting the JWT decode key based on the **UNVER-IFIED** contents of the token. Think carefully before using this functionality, in most cases you probably don't need it.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the unverified JWT.

The second argument is a dictionary containing the payload data of the unverified JWT.

The decorated function must return a *string* that is used to decode and verify the token.

# **encode\_key\_loader**(*callback: Callable*) → Callable

This decorator sets the callback function for dynamically setting the JWT encode key based on the tokens identity. Think carefully before using this functionality, in most cases you probably don't need it.

The decorated function must take **one** argument.

The argument is the identity used to create this JWT.

The decorated function must return a string which is the secrete key used to encode the JWT.

# **expired\_token\_loader**(*callback: Callable*) → Callable

This decorator sets the callback function for returning a custom response when an expired JWT is encountered.

The decorated function must take two arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return a Flask Response.

#### invalid\_token\_loader(callback: Callable) → Callable

This decorator sets the callback function for returning a custom response when an invalid JWT is encountered.

This decorator sets the callback function that will be used if an invalid JWT attempts to access a protected endpoint.

The decorated function must take **one** argument.

The argument is a string which contains the reason why a token is invalid.

The decorated function must return a Flask Response.

# $needs\_fresh\_token\_loader(callback: Callable) \rightarrow Callable$

This decorator sets the callback function for returning a custom response when a valid and non-fresh token is used on an endpoint that is marked as fresh=True.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return a Flask Response.

# $revoked\_token\_loader(callback: Callable) \rightarrow Callable$

This decorator sets the callback function for returning a custom response when a revoked token is encountered.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return a Flask Response.

#### token\_in\_blocklist\_loader(callback: Callable) → Callable

This decorator sets the callback function used to check if a JWT has been revoked.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must be return True if the token has been revoked, False otherwise.

### token\_verification\_failed\_loader(callback: Callable) → Callable

This decorator sets the callback function used to return a custom response when the claims verification check fails.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return a Flask Response.

#### token\_verification\_loader(callback: Callable) → Callable

This decorator sets the callback function used for custom verification of a valid JWT.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return True if the token is valid, or False otherwise.

# $unauthorized\_loader(callback: Callable) \rightarrow Callable$

This decorator sets the callback function used to return a custom response when no JWT is present.

The decorated function must take **one** argument.

The argument is a string that explains why the JWT could not be found.

The decorated function must return a Flask Response.

# user\_identity\_loader(callback: Callable) → Callable

This decorator sets the callback function used to convert an identity to a string when creating JWTs. This is useful for using objects (such as SQLAlchemy instances) as the identity when creating your tokens.

The decorated function must take **one** argument.

The argument is the identity that was used when creating a JWT.

The decorated function must return a string.

# user\_lookup\_loader(callback: Callable) → Callable

This decorator sets the callback function used to convert a JWT into a python object that can be used in a protected endpoint. This is useful for automatically loading a SQLAlchemy instance based on the contents of the JWT.

The object returned from this function can be accessed via current\_user or get\_current\_user()

The decorated function must take two arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function can return any python object, which can then be accessed in a protected endpoint. If an object cannot be loaded, for example if a user has been deleted from your database, None must be returned to indicate that an error occurred loading the user.

# user\_lookup\_error\_loader(callback: Callable) → Callable

This decorator sets the callback function used to return a custom response when loading a user via user\_loader() fails.

The decorated function must take **two** arguments.

The first argument is a dictionary containing the header data of the JWT.

The second argument is a dictionary containing the payload data of the JWT.

The decorated function must return a Flask Response.

## AFL.automation.APIServer.APIServer.LoggerFilter

## **Methods**

```
__init__(*filters)
```

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

## AFL.automation.APIServer.APIServer.MutableQueue

class AFL.automation.APIServer.APIServer.MutableQueue

Bases: object

Thread-safe, mutable queue

Unlike the standard library CPython queue, this class supportes positional inserts, deletions, and reordering. The tradeoff is performance for both reads and writes to the queue.

\_\_init\_\_()

## **Methods**

init()	
empty()	
<pre>get([loc, block, timeout])</pre>	Get next item from queue
<pre>iterationid()</pre>	
<pre>move(old_index[, new_index])</pre>	Move item in queue
put(item, loc)	Insert an item at the top of the queue
qsize()	
remove(loc)	Remove an item from the queue

```
__init__()
qsize()
iterationid()
empty()
```

#### AFL.automation.APIServer.APIServer.QueueDaemon

Bases: Thread

\_\_init\_\_(app, driver, task\_queue, history, debug=False, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

## **Methods**

init(app, driver, task_queue, history[,])	This constructor should always be called with keyword arguments.
<pre>check_if_paused()</pre>	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>mask_serialized_objs(package)</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

## **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

## **\_\_init\_\_**(app, driver, task\_queue, history, debug=False, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### terminate()

## check\_if\_paused()

## property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

## getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

## isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

## mask\_serialized\_objs(package)

## property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

## setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## AFL.automation.APIServer.APIServer.SMTPHandler

Bases: Handler

A handler class which sends an SMTP email for each logging event.

\_\_init\_\_(mailhost, fromaddr, toaddrs, subject, credentials=None, secure=None, timeout=5.0)
Initialize the handler.

Initialize the instance with the from and to addresses and subject line of the email. To specify a non-standard SMTP port, use the (host, port) tuple format for the mailhost argument. To specify authentication credentials, supply a (username, password) tuple for the credentials argument. To specify the use of a secure protocol (TLS), pass in a tuple for the secure argument. This will only be used when authentication credentials are supplied. The tuple will be either an empty tuple, or a single-value tuple with the name of a keyfile, or a 2-value tuple with the names of the keyfile and certificate file. (This tuple is passed to the *starttls* method). A timeout in seconds can be specified for the SMTP connection (the default is one second).

## **Methods**

init (mailhast framaddr taaddra subject)	Initialize the handler.
init(mailhost, fromaddr, toaddrs, subject)	
acquire()	Acquire the I/O thread lock.
addFilter(filter)	Add the specified filter to this handler.
close()	Tidy up any resources used by the handler.
createLock()	Acquire a thread lock for serializing access to the underlying I/O.
emit(record)	Emit a record.
filter(record)	Determine if a record is loggable by consulting all the filters.
flush()	Ensure all logging output has been flushed.
<pre>format(record)</pre>	Format the specified record.
<pre>getSubject(record)</pre>	Determine the subject for the email.
<pre>get_name()</pre>	
handle(record)	Conditionally emit the specified logging record.
handleError(record)	Handle errors which occur during an emit() call.
release()	Release the I/O thread lock.
removeFilter(filter)	Remove the specified filter from this handler.
setFormatter(fmt)	Set the formatter for this handler.
setLevel(level)	Set the logging level of this handler.
<pre>set_name(name)</pre>	

#### **Attributes**

name

\_\_init\_\_(mailhost, fromaddr, toaddrs, subject, credentials=None, secure=None, timeout=5.0)

Initialize the handler.

Initialize the instance with the from and to addresses and subject line of the email. To specify a non-standard SMTP port, use the (host, port) tuple format for the mailhost argument. To specify authentication credentials, supply a (username, password) tuple for the credentials argument. To specify the use of a secure protocol (TLS), pass in a tuple for the secure argument. This will only be used when authentication credentials are supplied. The tuple will be either an empty tuple, or a single-value tuple with the name of a keyfile, or a 2-value tuple with the names of the keyfile and certificate file. (This tuple is passed to the *starttls* method). A timeout in seconds can be specified for the SMTP connection (the default is one second).

### getSubject(record)

Determine the subject for the email.

If you want to specify a subject line which is record-dependent, override this method.

#### emit(record)

Emit a record.

Format the record and send it to the specified addressees.

#### acquire()

Acquire the I/O thread lock.

#### addFilter(filter)

Add the specified filter to this handler.

#### close()

Tidy up any resources used by the handler.

This version removes the handler from an internal map of handlers, \_handlers, which is used for handler lookup by name. Subclasses should ensure that this gets called from overridden close() methods.

## createLock()

Acquire a thread lock for serializing access to the underlying I/O.

#### filter(record)

Determine if a record is loggable by consulting all the filters.

The default is to allow the record to be logged; any filter can veto this and the record is then dropped. Returns a zero value if a record is to be dropped, else non-zero.

Changed in version 3.2: Allow filters to be just callables.

## flush()

Ensure all logging output has been flushed.

This version does nothing and is intended to be implemented by subclasses.

#### format(record)

Format the specified record.

If a formatter is set, use it. Otherwise, use the default formatter for the module.

#### get\_name()

## handle(record)

Conditionally emit the specified logging record.

Emission depends on filters which may have been added to the handler. Wrap the actual emission of the record with acquisition/release of the I/O thread lock. Returns whether the filter passed the record for emission.

#### handleError(record)

Handle errors which occur during an emit() call.

This method should be called from handlers when an exception is encountered during an emit() call. If raiseExceptions is false, exceptions get silently ignored. This is what is mostly wanted for a logging system - most users will not care about errors in the logging system, they are more interested in application errors. You could, however, replace this with a custom handler if you wish. The record which was being processed is passed in to this method.

#### property name

## release()

Release the I/O thread lock.

## removeFilter(filter)

Remove the specified filter from this handler.

#### setFormatter(fmt)

Set the formatter for this handler.

#### setLevel(level)

Set the logging level of this handler. level must be an int or a str.

set\_name(name)

#### AFL.automation.APIServer.APIServer.ServiceInfo

## class AFL.automation.APIServer.APIServer.ServiceInfo

Bases: RecordUpdateListener

Service information.

Constructor parameters are as follows:

- type: fully qualified service type name
- *name*: fully qualified service name
- port: port that the service runs on
- weight: weight of the service
- priority: priority of the service
- *properties*: dictionary of properties (or a bytes object holding the contents of the *text* field). converted to str and then encoded to bytes using UTF-8. Keys with *None* values are converted to value-less attributes.
- server: fully qualified name for service host (defaults to name)
- host ttl: ttl used for A/SRV records
- other ttl: ttl used for PTR/TXT records

- addresses and parsed\_addresses: List of IP addresses (either as bytes, network byte order, or in parsed form as text; at most one of those parameters can be provided)
- interface\_index: scope\_id or zone\_id for IPv6 link-local addresses i.e. an identifier of the interface where the peer is connected to

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

## **Methods**

	dresses matching IP version.
The state of the s	e e e e e e e e e e e e e e e e e e e
	e
async_clear_cache() Clear th	
	ne cache for this service info.
,	s true if the service could be discovered on the k, and updates this object with details discov-
async_update_records(zc, now, records) Updates	s service information from a DNS record.
async_update_records_complete() Called handler	when a record update has completed for all s.
	task waits for a given number of milliseconds notified.
dns_addresses([override_ttl, version]) Return	matching DNSAddress from ServiceInfo.
<pre>dns_nsec(missing_types[, override_ttl])</pre> Return	DNSNsec from ServiceInfo.
<pre>dns_pointer([override_ttl])</pre> Return	DNSPointer from ServiceInfo.
<pre>dns_service([override_ttl])</pre> Return	DNSService from ServiceInfo.
<pre>dns_text([override_ttl])</pre> Return	DNSText from ServiceInfo.
å	set of address records and NSEC records for esent record types.
get_name() Name a	ccessor
<pre>ip_addresses_by_version(version)</pre> List ip_	address objects matching IP version.
load_from_cache(zc[, now]) Populat	te the service info from the cache.
parsed_addresses([version]) List add	dresses in their parsed string form.
that IP	lent to parsed_addresses, with the exception v6 Link-Local addresses are qualified with rface_index> when available
	s true if the service could be discovered on the k, and updates this object with details discov-
set_server_if_missing() Set the	server if it is missing.
<pre>update_record(zc, now, record)</pre> Update	a single record.

## **Attributes**

```
host_ttl
interface_index
key
other\_ttl
port
priority
server
server_key
text
type
weight
addresses
                                                 IPv4 addresses of this service.
decoded_properties
                                                 Return properties as strings.
                                                 The name of the service.
name
                                                 Return properties as bytes.
properties
```

```
host_ttl
interface_index
key
other_ttl
port
priority
server
server_key
text
type
weight
__init__(*args, **kwargs)
```

#### addresses

IPv4 addresses of this service.

Only IPv4 addresses are returned for backward compatibility. Use addresses\_by\_version() or parsed\_addresses() to include IPv6 addresses as well.

## addresses\_by\_version(version: IPVersion)

List addresses matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

## async\_clear\_cache()

Clear the cache for this service info.

```
async_request(zc: Zeroconf, timeout: float, question\_type: DNSQuestionType | None = None, addr: str | None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

This method will be run in the event loop.

Passing addr and port is optional, and will default to the mDNS multicast address and port. This is useful for directing requests to a specific host that may be able to respond across subnets.

#### **Parameters**

- **zc** Zeroconf instance
- timeout time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- port port to send the request to

```
async_update_records(zc: Zeroconf, now: float_, records: list[RecordUpdate])
```

Updates service information from a DNS record.

This method will be run in the event loop.

#### async\_update\_records\_complete()

Called when a record update has completed for all handlers.

At this point the cache will have the new records.

This method will be run in the event loop.

```
async_wait(timeout: float, loop: AbstractEventLoop | None = None) <math>\rightarrow None
```

Calling task waits for a given number of milliseconds or until notified.

## decoded\_properties

Return properties as strings.

```
dns\_addresses(override\_ttl: int\_ | None = None, version: IPVersion = IPVersion.All) \rightarrow list[DNSAddress]
Return matching DNSAddress from ServiceInfo.
```

```
dns\_nsec(missing\_types: list[int], override\_ttl: int\_ | None = None) \rightarrow DNSNsec
```

Return DNSNsec from ServiceInfo.

```
dns_pointer(override_ttl: int_| None = None) \rightarrow DNSPointer
```

Return DNSPointer from ServiceInfo.

## $dns_service(override_ttl: int_ | None = None) \rightarrow DNSService$

Return DNSService from ServiceInfo.

```
dns_text(override_ttl: int_| None = None) \rightarrow DNSText
```

Return DNSText from ServiceInfo.

## get\_address\_and\_nsec\_records( $override\_ttl: int\_ \mid None = None$ ) $\rightarrow set[DNSRecord]$

Build a set of address records and NSEC records for non-present record types.

```
get_name() \rightarrow str
```

Name accessor

## ip\_addresses\_by\_version(version: IPVersion)

List ip\_address objects matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
load_from_cache(zc: Zeroconf, now: float_| None = None) \rightarrow bool
```

Populate the service info from the cache.

This method is designed to be threadsafe.

#### name

The name of the service.

```
parsed\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

List addresses in their parsed string form.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
parsed\_scoped\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

Equivalent to parsed\_addresses, with the exception that IPv6 Link-Local addresses are qualified with %<interface\_index> when available

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

#### properties

Return properties as bytes.

```
request(zc: Zeroconf, timeout: float, question_type: DNSQuestionType | None = None, addr: str \mid None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_request* cannot be completed.

#### **Parameters**

• **zc** – Zeroconf instance

- **timeout** time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- port port to send the request to

```
set_server_if_missing() → None
```

Set the server if it is missing.

This function is for backwards compatibility.

```
\mathbf{update\_record}(\mathit{zc} \colon \mathsf{Zeroconf}, \mathit{now} \colon \mathit{float}, \mathit{record} \colon \mathit{DNSRecord}) \to \mathsf{None}
```

Update a single record.

This method is deprecated and will be removed in a future version. update\_records should be implemented instead.

#### AFL.automation.APIServer.APIServer.Zeroconf

```
class AFL.automation.APIServer.APIServer.Zeroconf(interfaces: Sequence[str | int | tuple[str, int, int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False)
```

Bases: QuietLogger

Implementation of Zeroconf Multicast DNS Service Discovery

Supports registration, unregistration, queries and browsing.

```
__init__(interfaces: Sequence[str | int | tuple[tuple[str, int, int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

#### **Parameters**

• **interfaces** – **InterfaceChoice** or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

## Methods

init([interfaces, unicast, ip_version,])	Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.
add list on a religion on assertion)	
add_listener(listener, question)	Adds a listener for a given question.
add_service_listener(type_, listener)	Adds a listener for a particular service type.
async_add_listener(listener, question)	Adds a listener for a given question.
<pre>async_check_service(info, allow_name_change)</pre>	Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique.
<pre>async_get_service_info(type_, name[,])</pre>	Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.
<pre>async_notify_all()</pre>	Schedule an async_notify_all.
<pre>async_register_service(info[, ttl,])</pre>	Registers service information to the network with a default TTL.
<pre>async_remove_listener(listener)</pre>	Removes a listener.
<pre>async_send(out[, addr, port, v6_flow_scope,])</pre>	Sends an outgoing packet.
<pre>async_unregister_all_services()</pre>	Unregister all registered services.
async_unregister_service(info)	Unregister a service.
async_update_service(info)	Registers service information to the network with a default TTL.
<pre>async_wait(timeout)</pre>	Calling task waits for a given number of milliseconds or until notified.
<pre>async_wait_for_start([timeout])</pre>	Wait for start up for actions that require a running Zeroconf instance.
close()	Ends the background threads, and prevent this instance from servicing further queries.
<pre>generate_service_broadcast(info, ttl[,])</pre>	Generate a broadcast to announce a service.
<pre>generate_service_query(info)</pre>	Generate a query to lookup a service.
<pre>generate_unregister_all_services()</pre>	Generate a DNSOutgoing goodbye for all services and remove them from the registry.
<pre>get_service_info(type_, name[, timeout,])</pre>	Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.
log_exception_debug(*logger_data)	
<pre>log_exception_once(exc, *args)</pre>	
<pre>log_exception_warning(*logger_data)</pre>	
<pre>log_warning_once(*args)</pre>	
notify_all()	Notifies all waiting threads and notify listeners.
register_service(info[, ttl,])	Registers service information to the network with a default TTL.
remove_all_service_listeners()	Removes a listener from the set that is currently listening.
remove_listener(listener)	Removes a listener.
remove_service_listener(listener)	Removes a listener from the set that is currently listening.
<pre>send(out[, addr, port, v6_flow_scope, transport])</pre>	Sends an outgoing packet threadsafe.
	continues on next page

continues on next page

Table 4 – continued from previous page

start()	Start Zeroconf.
<pre>unregister_all_services()</pre>	Unregister all registered services.
unregister_service(info)	Unregister a service.
update_service(info)	Registers service information to the network with a default TTL.

## **Attributes**

listeners	
started	Check if the instance has started.

**\_\_init\_\_**(interfaces: Sequence[str | int | tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip\_version: IPVersion | None = None, apple\_p2p: bool = False)  $\rightarrow$  None

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

#### **Parameters**

• **interfaces** – **InterfaceChoice** or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

## property started: bool

Check if the instance has started.

 $start() \rightarrow None$ 

Start Zeroconf.

```
async async_wait_for_start(timeout: float = 9) \rightarrow None
```

Wait for start up for actions that require a running Zeroconf instance.

Throws NotRunningException if the instance is not running or could not be started.

property listeners: set[RecordUpdateListener]

async async\_wait(timeout: float)  $\rightarrow$  None

Calling task waits for a given number of milliseconds or until notified.

 $notify_all() \rightarrow None$ 

Notifies all waiting threads and notify listeners.

```
async_notify_all() \rightarrow None
```

Schedule an async\_notify\_all.

```
get_service_info(type_: str, name: str, timeout: int = 3000, question_type: DNSQuestionType | None = None) \rightarrow ServiceInfo | None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

#### **Parameters**

- type fully qualified service type name
- name the name of the service
- timeout milliseconds to wait for a response
- **question\_type** The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

```
add\_service\_listener(type\_: str, listener: ServiceListener) \rightarrow None
```

Adds a listener for a particular service type. This object will then have its add\_service and remove\_service methods called when services of that type become available and unavailable.

```
remove\_service\_listener(listener: ServiceListener) \rightarrow None
```

Removes a listener from the set that is currently listening.

```
remove_all_service_listeners() → None
```

Removes a listener from the set that is currently listening.

```
register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating responders: bool = False, strict: bool = True) \rightarrow None
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *register service* cannot be completed.

```
async async_register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating_responders: bool = False, strict: bool = True) \rightarrow Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

```
update_service(info: ServiceInfo) \rightarrow None
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_update\_service* cannot be completed.

```
async async_update_service(info: ServiceInfo) → Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

```
async async_get_service_info(type_: str, name: str, timeout: int = 3000, question_type: DNSQuestionType | None = None) <math>\rightarrow AsyncServiceInfo | None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

#### **Parameters**

- type fully qualified service type name
- name the name of the service
- timeout milliseconds to wait for a response
- question\_type The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

```
generate\_service\_broadcast(info: ServiceInfo, ttl: int | None, broadcast\_addresses: bool = True) \rightarrow DNSOutgoing
```

Generate a broadcast to announce a service.

```
\textbf{generate\_service\_query}(\textit{info}: ServiceInfo}) \rightarrow DNSOutgoing
```

Generate a query to lookup a service.

```
unregister_service(info: ServiceInfo) \rightarrow None
```

Unregister a service.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_unregister\_service* cannot be completed.

```
async async_unregister_service(info: ServiceInfo) → Awaitable
```

Unregister a service.

```
\texttt{generate\_unregister\_all\_services()} \rightarrow DNSOutgoing \mid None
```

Generate a DNSOutgoing goodbye for all services and remove them from the registry.

```
async async_unregister_all_services() \rightarrow None
```

Unregister all registered services.

Unlike async\_register\_service and async\_unregister\_service, this method does not return a future and is always expected to be awaited since its only called at shutdown.

```
\textbf{unregister\_all\_services()} \rightarrow None
```

Unregister all registered services.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_unregister\_all\_services* cannot be completed.

```
async async_check_service(info: ServiceInfo, allow_name_change: bool, cooperating_responders: bool = False, strict: bool = True) \rightarrow None
```

Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique.

```
 \textbf{add\_listener}(\textit{listener: RecordUpdateListener, question: DNSQuestion} \mid \textit{list[DNSQuestion]} \mid \textit{None}) \rightarrow \\ \text{None}
```

Adds a listener for a given question. The listener will have its update\_record method called when information is available to answer the question(s).

This function is threadsafe

```
remove_listener(listener: RecordUpdateListener) \rightarrow None
           Removes a listener.
           This function is threadsafe
     async_add_listener(listener: RecordUpdateListener, question: DNSQuestion | list[DNSQuestion] | None)
           Adds a listener for a given question. The listener will have its update_record method called when informa-
           tion is available to answer the question(s).
           This function is not threadsafe and must be called in the eventloop.
     async\_remove\_listener(listener: RecordUpdateListener) \rightarrow None
           Removes a listener.
           This function is not threadsafe and must be called in the eventloop.
     send(out: DNSOutgoing, addr: str | None = None, port: int = 5353, v6_flow_scope: tuple[()] | tuple[int, int]
            = (), transport: \_WrappedTransport | None = None) \rightarrow None
           Sends an outgoing packet threadsafe.
     async_send(out: DNSOutgoing, addr: str | None = None, port: int = 5353, v6_flow_scope: tuple[()] |
                   tuple[int, int] = (), transport: \_WrappedTransport | None = None) \rightarrow None
           Sends an outgoing packet.
     close() \rightarrow None
           Ends the background threads, and prevent this instance from servicing further queries.
           This method is idempotent and irreversible.
     classmethod log_exception_debug(*logger\_data: Any) \rightarrow None
     classmethod log_exception_once(exc: Exception, *args: Any) \rightarrow None
     classmethod log_exception_warning(*logger\_data: Any) \rightarrow None
     classmethod log_warning_once(*args: Any) \rightarrow None
class AFL.automation.APIServer.APIServer(name, data=None, experiment='Development',
                                                                contact='tbm@nist.gov',
                                                                index_template='index.html',
                                                                new index template='index-new.html',
                                                                plot template='simple-bokeh.html')
     __init__(name, data=None, experiment='Development', contact='tbm@nist.gov',
                 index_template='index.html', new_index_template='index-new.html',
                 plot_template='simple-bokeh.html')
     create_queue(driver, add_unqueued=True)
     reset_queue_daemon(driver=None)
     advertise_zeroconf(**kwargs)
     run(**kwargs)
     run_threaded(start_thread=True, **kwargs)
     add_standard_routes()
```

```
get_info()
     Live, status page of the robot
get_quickbar()
     Return the functions, params, and defaults to be shown in this server's quickbar
is_server_live()
get_unqueued_commands()
get_queued_commands()
add_unqueued_routes()
query_driver()
init_logging(toaddrs=None)
index()
     Live, status page of the robot
index_new()
     Live, status page of the robot
webapp()
     Live, status page of the robot
render_unqueued(func, kwargs_add, **kwargs)
     Convert an unqueued return item into web-suitable output
send_1d_plot(result, multi=False, **kwargs)
send_array_as_jpg(array, log_image=False, max_val=None, fillna=0.0, **kwargs)
queue_state()
driver_status()
get_queue()
get_queue_iteration()
deposit_obj()
     Store an object named obj in the driver's dropbox If a uuid is provided, the object will be stored with that
     uuid Otherwise, a new uuid will be generated. In either case, the uuid will be returned to the client.
retrieve_obj()
     Retrieve an object from the driver's dropbox unid specifies the object to retrieve delete specifies whether
     to delete the object after retrieval
set_driver_object()
get_driver_object()
enqueue()
reorder_queue()
remove_items()
```

```
remove_item()
move_item()
clear_queue()
clear_history()
debug()
pause()
halt()
init()
login()
get_server_time()
login_test()
```

#### AFL.automation.APIServer.Client

#### **Classes**

<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
ServerDiscovery()	ServerDiscovery class

## AFL.automation.APIServer.Client.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
```

#### **Methods**

```
__init__([ip, port, username, interactive])

clear_history()

clear_queue()
```

continues on next page

Table 5 – continued from previous page

	d from previous page
debug(state)	
<pre>deposit_obj(obj[, uid])</pre>	Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object under if not specified, a new uuid will be generated
<pre>driver_status()</pre>	
enqueue([interactive])	
enqueued_base(**kwargs)	
<pre>from_server_name(server_name, **kwargs)</pre>	
<pre>get_config(name[, print_console, interactive])</pre>	
<pre>get_driver_object(name)</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_queue()</pre>	
<pre>get_queued_commands([inherit_commands])</pre>	
<pre>get_quickbar()</pre>	
<pre>get_server_time()</pre>	
<pre>get_unqueued_commands([inherit_commands])</pre>	
halt()	
logged_in()	
<pre>login(username[, populate_commands])</pre>	
<pre>move_item(uuid, pos)</pre>	
pause(state)	
query_driver(**kwargs)	
queue_state()	
remove_item(uuid)	
reset_queue_daemon()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete the object after retrieving
server_cmd(cmd, **kwargs)	
	continues on next nage

continues on next page

## Table 5 – continued from previous page

```
set_config([interactive])
 set_driver_object(**kw)
 set_object([serialize])
 unqueued_base(**kwargs)
 wait([target_uuid, interval, for_history, ...])
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
```

## AFL.automation.APIServer.Client.ServerDiscovery

```
class AFL.automation.APIServer.Client.ServerDiscovery
```

Bases: object

ServerDiscovery class

This class is used to discover AFL servers on the network using zeroconf requests that match a particular specification.

\_\_init\_\_()

## **Methods**

init()	
<pre>aio_find_server_by_name(service_name)</pre>	
<pre>discover_server_by_name(service_name)</pre>	Does a zeroconf request to find a named AFL-automation server on the network.
<pre>find_server_by_name(service_name)</pre>	Disambiguator for either matching or discovering a specific server by name
<pre>find_server_by_partial_name(service_name)</pre>	Looks through the registry of discovered services for a partial name match.
<pre>find_server_by_property_match(property_name)</pre>	Looks through the registry of discovered services for a partial match in a property string.
<pre>get_service_info(zeroconf, service_type, name)</pre>	
<pre>manage_service_info_to_list(zeroconf,)</pre>	
<pre>match_server_by_name(service_name)</pre>	Looks through the registry of discovered services for an exact name match.
<pre>on_service_state_change(zeroconf,)</pre>	
sa_aio_discover_server_by_name(service_name	Does a zeroconf request to find a named AFL-automation server on the network.
<pre>sa_discover_server_by_name(service_name)</pre>	Does a zeroconf request to find a named AFL-automation server on the network.

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

```
async classmethod sa_aio_discover_server_by_name(service_name)
```

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## classmethod sa\_discover\_server\_by\_name(service\_name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## find\_server\_by\_name(service\_name)

Disambiguator for either matching or discovering a specific server by name

```
match_server_by_name(service_name)
```

Looks through the registry of discovered services for an exact name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

```
find_server_by_partial_name(service_name)
```

Looks through the registry of discovered services for a partial name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

```
find_server_by_property_match(property_name, property_value)
```

Looks through the registry of discovered services for a partial match in a property string. Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
```

```
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
set_driver_object(**kw)
get_driver_object(name)
deposit_obj(obj, uid=None)
     Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
     under if not specified, a new uuid will be generated
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
     the object after retrieving
set_object(serialize=True, **kw)
get_object(name, serialize=True)
```

## AFL.automation.APIServer.Driver

## **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
makeRegistrar()	
sqrt(x,/)	Return the square root of x.

## AFL.automation.APIServer.Driver.ceil

```
AFL.automation.APIServer.Driver.ceil(x,/)
Return the ceiling of x as an Integral.
This is the smallest integer >= x.
```

## AFL.automation.APIServer.Driver.listify

AFL.automation.APIServer.Driver.listify(obj)

## AFL.automation.APIServer.Driver.makeRegistrar

AFL.automation.APIServer.Driver.makeRegistrar()

## AFL.automation.APIServer.Driver.sqrt

AFL.automation.APIServer.Driver. $\mathbf{sqrt}(x,/)$ Return the square root of x.

## **Classes**

```
      Driver(name[, defaults, overrides])

      PersistentConfig(path[, defaults, ...])
      A dictionary-like class that serializes changes to disk
```

## AFL.automation.APIServer.Driver.Driver

```
class AFL.automation.APIServer.Driver.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

## **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

## AFL.automation.APIServer.Driver.PersistentConfig

Bases: MutableMapping

A dictionary-like class that serializes changes to disk

This class provides dictionary-like setters and getters (e.g., [] and .update()) but, by default, all modifications are written into a file in json format with the root keys as timestamps. Modifications can be blocked by locking the config, and writing to disk can be disabled by setting the appropriate member attributes (see constructor). On instantiation, if provided with a previously saved configuration file, PersistentConfig will load the file's contents into memory and use the more recent configuration.

\_\_init\_\_(path, defaults=None, overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

#### Constructor

#### **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

## **Methods**

init(path[, defaults, overrides, lock,])		Constructor
clear()		
<pre>get(k[,d])</pre>		
<pre>get_historical_values(key[, vert_to_datetime])</pre>	con-	Convenience method for gathering historical values of a parameter
<pre>items()</pre>		
keys()		
<i>pop</i> (k[,d])		If key is not found, d is returned if given, otherwise KeyError is raised.
<pre>popitem()</pre>		as a 2-tuple; but raise KeyError if D is empty.
<pre>revert([nth, datetime_key])</pre>		Revert config to a historical config
setdefault(k[,d])		
toJSON()		Serialize the config to json
<pre>update(update_dict)</pre>		Update several values in config at once
values()		

\_\_init\_\_(path, defaults=None, overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

#### Constructor

#### **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

```
__getitem__(key)
```

Dictionary-like getter via config["param"]

```
__setitem__(key, value)
```

Dictionary-like setter via config["param"]=value

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

```
toJSON()
           Serialize the config to json
      update(update dict)
           Update several values in config at once
           Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).
      revert(nth=None, datetime key=None)
           Revert config to a historical config
                Parameters
                    • nth (int, optional*) – Integer index of historical value to revert to. Can be negative to
                      count from end of history. Note that -1 will correspond to the current config.
                    • datetime_key (str, optional) - datetime formatted string as defined by date-
                      time_key_format
      get_historical_values(key, convert_to_datetime=False)
           Convenience method for gathering historical values of a parameter
      clear() \rightarrow None. Remove all items from D.
      get(k[,d]) \rightarrow D[k] if k in D, else d. d defaults to None.
      items() \rightarrow a set-like object providing a view on D's items
      keys() \rightarrow a set-like object providing a view on D's keys
      pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If key is not found, d is returned if given, otherwise KeyError is raised.
      popitem() \rightarrow (k, v), remove and return some (key, value) pair
           as a 2-tuple; but raise KeyError if D is empty.
      setdefault(k[, d]) \rightarrow D.get(k,d), also set D[k]=d if k not in D
      values() \rightarrow an object providing a view on D's values
AFL.automation.APIServer.Driver.makeRegistrar()
class AFL.automation.APIServer.Driver.Driver(name, defaults=None, overrides=None)
      unqueued()
      queued()
      quickbar()
      __init__(name, defaults=None, overrides=None)
      classmethod gather_defaults()
           Gather all inherited static class-level dictionaries called default.
      set_config(**kwargs)
      get_config(name, print_console=False)
      get_configs(print_console=False)
```

```
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) - The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
```

- **obj** (*object*) The object to store in the dropbox
- **uid** (*str*) The uuid to store the object under

## AFL.automation.APIServer.DummyDriver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x, l)	Return the square root of x.

## AFL.automation.APIServer.DummyDriver.ceil

```
AFL.automation.APIServer.DummyDriver.ceil(x,/)
Return the ceiling of x as an Integral.
This is the smallest integer >= x.
```

## AFL.automation.APIServer.DummyDriver.listify

```
{\tt AFL.automation.APIServer.DummyDriver.listify} (\it obj)
```

## AFL.automation.APIServer.DummyDriver.sqrt

```
AFL.automation.APIServer.DummyDriver.sqrt(x,/)
Return the square root of x.
```

## Classes

```
Driver(name[, defaults, overrides])

DummyDriver([name, overrides])
```

## AFL.automation.APIServer.DummyDriver.Driver

```
class AFL.automation.APIServer.DummyDriver.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

## **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.APIServer.DummyDriver.DummyDriver

class AFL.automation.APIServer.DummyDriver.DummyDriver(name=None, overrides=None)
 Bases: Driver
 \_\_init\_\_(name=None, overrides=None)

```
_init__([name, overrides])
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
dummy_reset_tank_levels([rinse1,
                                          rinse2,
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
get_config(name[, print_console])
{\tt get\_configs}([print\_console])
get_object(name[, serialize])
get_sample()
how_many(**kwargs)
loadSample([cellname, sampleVolume])
                                                   Executed after each call to execute
post_execute(**kwargs)
pre_execute(**kwargs)
                                                   Executed before each call to execute
queued()
quickbar()
quickbar_test([text_field, int_field, ...])
quickbar_test2()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                   Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
test_command1([kwarg1, kwarg2])
                                                   A test command with positional and keyword param-
test_command2([kwarg1, kwarg2, kwarg3])
                                                   A test command with positional and keyword param-
test_command_sets_data([kwarg1,
                                         kwarg2,
                                                   A test command with positional and keyword param-
kwarg3])
                                                   eters
test_image(**kwargs)
test_plot(**kwargs)
```

6.2.  $Modules_{ed()}$ 

#### **Attributes**

```
defaults
```

```
defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
__init__(name=None, overrides=None)
status()
test_command1(kwarg1=None, kwarg2=True)
    A test command with positional and keyword parameters
test_command2(kwarg1=False, kwarg2=False, kwarg3=True)
    A test command with positional and keyword parameters
test\_command\_sets\_data(kwarg1=False, kwarg2=False, kwarg3=True)
    A test command with positional and keyword parameters
how_many(**kwargs)
test_plot(**kwargs)
test_image(**kwargs)
quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
quickbar_test2()
dummy_reset_tank_levels(rinse1=950, rinse2=950, waste=0)
loadSample(cellname='cell', sampleVolume=0)
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
    Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
```

```
post_execute(**kwargs)
          Executed after each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     pre_execute(**kwargs)
          Executed before each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     queued()
     quickbar()
     reset_sample()
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                  uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
               Parameters
                   • data (dict) – Dictionary of data to store in the driver object
                   • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
class AFL.automation.APIServer.DummyDriver.DummyDriver(name=None, overrides=None)
     defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
     __init__(name=None, overrides=None)
     status()
     test_command1(kwarg1=None, kwarg2=True)
          A test command with positional and keyword parameters
     test_command2(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     test_command_sets_data(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     how_many(**kwargs)
```

```
test_plot(**kwargs)
test_image(**kwargs)
quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
quickbar_test2()
dummy_reset_tank_levels(rinsel=950, rinse2=950, waste=0)
loadSample(cellname='cell', sampleVolume=0)
```

## AFL.automation.APIServer.DummyOT2Driver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x, l)	Return the square root of x.

## AFL.automation.APIServer.DummyOT2Driver.ceil

```
AFL.automation.APIServer.DummyOT2Driver.ceil(x,/)
Return the ceiling of x as an Integral.
This is the smallest integer >= x.
```

## AFL.automation.APIServer.DummyOT2Driver.listify

AFL.automation.APIServer.DummyOT2Driver.listify(obj)

# AFL.automation.APIServer.DummyOT2Driver.sqrt

```
AFL.automation.APIServer.DummyOT2Driver.sqrt(x,/)
Return the square root of x.
```

#### **Classes**

```
Driver(name[, defaults, overrides])

DummyDriver([name, overrides])
```

# ${\bf AFL.} automation. {\bf APIS} erver. {\bf DummyOT2Driver. Driver}$

```
class AFL.automation.APIServer.DummyOT2Driver.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

# **Methods**

deposit_obj(obj[, uid]) Store an object in the dropbox   execute(**kwargs) Gather all inherited static class-level dictionaries called default.   get_config(name[, print_console]) get_configs([print_console])   get_object(name[, serialize]) get_sample()   post_execute(**kwargs) Executed after each call to execute   pre_execute(**kwargs) Executed before each call to execute   queued() quickbar()   reset_sample() retrieve_obj(uid[, delete])   set_config(**kwargs) Set data in the DataPacket object   set_object([serialized]) set_sample(sample_name[, sample_uuid])   status()	init(name[, defaults, overrides])	
<pre>gather_defaults()</pre>	<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
called default.  get_config(name[, print_console])  get_object(name[, serialize])  get_sample()  post_execute(**kwargs)	execute(**kwargs)	
<pre>get_configs([print_console])  get_object(name[, serialize])  get_sample()  post_execute(**kwargs)</pre>	<pre>gather_defaults()</pre>	
<pre>get_object(name[, serialize]) get_sample()  post_execute(**kwargs)</pre>	<pre>get_config(name[, print_console])</pre>	
<pre>get_sample()  post_execute(**kwargs)</pre>	<pre>get_configs([print_console])</pre>	
post_execute(**kwargs) pre_execute(**kwargs) pre_execute(**kwargs) Executed after each call to execute Executed before each call to execute  queued()  quickbar()  reset_sample()  retrieve_obj(uid[, delete]) set_config(**kwargs)  set_data(data) set_data(data) set_object([serialized])  set_sample(sample_name[, sample_uuid])	<pre>get_object(name[, serialize])</pre>	
pre_execute(**kwargs)  queued()  quickbar()  reset_sample()  retrieve_obj(uid[, delete])  set_config(**kwargs)  set_data(data)  set_object([serialized])  set_sample(sample_name[, sample_uuid])  Executed before each call to execute  execute  before each call to execute  Retrieve an object from the dropbox  Set data in the DataPacket object	<pre>get_sample()</pre>	
<pre>queued() quickbar()  reset_sample()  retrieve_obj(uid[, delete]) Retrieve an object from the dropbox  set_config(**kwargs)  set_data(data) Set data in the DataPacket object  set_object([serialized])  set_sample(sample_name[, sample_uuid])</pre>	<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>quickbar()  reset_sample()  retrieve_obj(uid[, delete]) Retrieve an object from the dropbox  set_config(**kwargs)  set_data(data) Set data in the DataPacket object  set_object([serialized])  set_sample(sample_name[, sample_uuid])</pre>		Executed before each call to execute
reset_sample()  retrieve_obj(uid[, delete]) Retrieve an object from the dropbox  set_config(**kwargs)  set_data(data) Set data in the DataPacket object  set_object([serialized])  set_sample(sample_name[, sample_uuid])	queued()	
<pre>retrieve_obj(uid[, delete])</pre>	quickbar()	
<pre>set_config(**kwargs)  set_data(data)</pre>	reset_sample()	
<pre>set_data(data) set_object([serialized]) set_sample(sample_name[, sample_uuid])</pre> Set data in the DataPacket object	- · · · - · · · · · · · · · · · · · · ·	Retrieve an object from the dropbox
<pre>set_object([serialized]) set_sample(sample_name[, sample_uuid])</pre>		
<pre>set_sample(sample_name[, sample_uuid])</pre>		Set data in the DataPacket object
	· · · · · · · · · · · · · · · · · · ·	
status()		
unqueued()	unqueued()	

unqueued()
queued()
quickbar()

```
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) - Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.APIServer.DummyOT2Driver.DummyDriver

```
class AFL.automation.APIServer.DummyOT2Driver.DummyDriver(name=None, overrides=None)
    Bases: Driver
    __init__(name=None, overrides=None)
```

#### **Methods**

```
__init__([name, overrides])
                                                   Store an object in the dropbox
deposit_obj(obj[, uid])
dummy_reset_tank_levels([rinsel,
                                          rinse2,
waste])
execute(**kwargs)
                                                   Gather all inherited static class-level dictionaries
gather_defaults()
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_prep_target(**kwargs)
get_sample()
how_many(**kwargs)
loadSample([cellname, sampleVolume])
                                                   Executed after each call to execute
post_execute(**kwargs)
                                                   Executed before each call to execute
pre_execute(**kwargs)
queued()
quickbar()
quickbar_test([text_field, int_field, ...])
quickbar_test2()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                   Set data in the DataPacket object
set_object([serialized])
```

continues on next page

Table 6 – continued from previous page

```
      set_sample(sample_name[, sample_uuid])

      status()

      test_command1([kwarg1, kwarg2])
      A test command with positional and keyword parameters

      test_command2([kwarg1, kwarg2, kwarg3])
      A test command with positional and keyword parameters

      test_command_sets_data([kwarg1, kwarg2, kwarg3])
      A test command with positional and keyword parameters

      test_image(**kwargs)
      test_plot(**kwargs)

      unqueued()
      unqueued()
```

### **Attributes**

defaults

```
defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
__init__(name=None, overrides=None)
status()
execute(**kwargs)
get_prep_target(**kwargs)
test_command1(kwarg1=None, kwarg2=True)
     A test command with positional and keyword parameters
\textbf{test\_command2}(\textit{kwarg1} = \textit{False}, \textit{kwarg2} = \textit{False}, \textit{kwarg3} = \textit{True})
     A test command with positional and keyword parameters
test_command_sets_data(kwarg1=False, kwarg2=False, kwarg3=True)
     A test command with positional and keyword parameters
how_many(**kwargs)
test_plot(**kwargs)
test_image(**kwargs)
quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
quickbar_test2()
dummy_reset_tank_levels(rinse1=950, rinse2=950, waste=0)
```

```
loadSample(cellname='cell', sampleVolume=0)
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
```

```
unqueued()
class AFL.automation.APIServer.DummyOT2Driver.DummyDriver(name=None, overrides=None)
     defaults = {'density of water': 1.0, 'speed of light': 300000000.0}
     __init__(name=None, overrides=None)
     status()
     execute(**kwargs)
     get_prep_target(**kwargs)
     test_command1(kwarg1=None, kwarg2=True)
          A test command with positional and keyword parameters
     test_command2(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     test_command_sets_data(kwarg1=False, kwarg2=False, kwarg3=True)
          A test command with positional and keyword parameters
     how_many(**kwargs)
     test_plot(**kwargs)
     test_image(**kwargs)
     quickbar_test(text_field='Three', int_field=3, float_field=3.14, bool_field=True)
     quickbar_test2()
     dummy_reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     loadSample(cellname='cell', sampleVolume=0)
AFL.automation.APIServer.LoggerFilter
```

#### **Classes**

```
LoggerFilter(*filters)
```

## AFL.automation.APIServer.LoggerFilter.LoggerFilter

```
class AFL.automation.APIServer.LoggerFilter.LoggerFilter(*filters)
     Bases: object
     __init__(*filters)
```

```
__init__(*filters)

__init__(*filters)

class AFL.automation.APIServer.LoggerFilter.LoggerFilter(*filters)

__init__(*filters)
```

## AFL.automation.APIServer.QueueDaemon

## **Functions**

```
is_serialized(obj)
```

# AFL.automation.APIServer.QueueDaemon.is\_serialized

AFL.automation.APIServer.QueueDaemon.is\_serialized(obj)

# **Classes**

DataTrashcan()	A DataPacket implementation <i>for testing only</i> that takes all its data and simply throws it away.
QueueDaemon(app, driver, task_queue, history)	

#### AFL.automation.APIServer.QueueDaemon.DataTrashcan

```
class AFL.automation.APIServer.QueueDaemon.DataTrashcan
    Bases: DataPacket
    A DataPacket implementation for testing only that takes all its data and simply throws it away.
    __init__()
```

init()	
add_array(*args, **kwargs)	Abstract method adding arrays that need special handling
clear()	
finalize()	
<pre>get(k[,d])</pre>	
<pre>items()</pre>	
keys()	
pop(k[,d])	If key is not found, d is returned if given, otherwise KeyError is raised.
<pre>popitem()</pre>	as a 2-tuple; but raise KeyError if D is empty.
reset()	Clears all transient data.
reset_sample()	
setdefault(k[,d])	
setupDefaults()	
transmit()	
update([E, ]**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(): $D[k] = v$
values()	

# **Attributes**

```
PROTECTED_SYSTEM_KEYS

transmit()
add_array(*args, **kwargs)
    Abstract method adding arrays that need special handling
PROTECTED_SAMPLE_KEYS = ['sample_name', 'sample_uuid', 'sample_composition', 'AL_components', 'AL_campaign_name', 'AL_uuid']
```

```
PROTECTED_SYSTEM_KEYS = ['driver_name', 'driver_config', 'platform_serial']
      __init__()
      clear() \rightarrow None. Remove all items from D.
      finalize()
      get(k[,d]) \rightarrow D[k] if k in D, else d. d defaults to None.
      items() \rightarrow a set-like object providing a view on D's items
      keys() \rightarrow a set-like object providing a view on D's keys
      pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If key is not found, d is returned if given, otherwise KeyError is raised.
      popitem() \rightarrow (k, v), remove and return some (key, value) pair
           as a 2-tuple; but raise KeyError if D is empty.
      reset()
           Clears all transient data.
      reset_sample()
      setdefault(k[, d]) \rightarrow D.get(k,d), also set D[k]=d if k not in D
      setupDefaults()
      update([E, ]^{**F}) \rightarrow 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
      values() \rightarrow an object providing a view on D's values
AFL.automation.APIServer.QueueDaemon.QueueDaemon
class AFL.automation.APIServer.QueueDaemon.QueueDaemon(app, driver, task_queue, history,
```

```
debug=False, data=None)
```

Bases: Thread

**\_\_init\_\_**(app, driver, task\_queue, history, debug=False, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init(app, driver, task_queue, history[,])	This constructor should always be called with keyword arguments.
<pre>check_if_paused()</pre>	
*	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>mask_serialized_objs(package)</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

**\_\_init\_\_**(app, driver, task\_queue, history, debug=False, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### terminate()

## check\_if\_paused()

## property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

## getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

## property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### mask\_serialized\_objs(package)

## property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

## property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

## setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

\_\_init\_\_(app, driver, task\_queue, history, debug=False, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### terminate()

#### check\_if\_paused()

#### mask\_serialized\_objs(package)

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

# 6.2.2 Instrument

The Instrument module contains drivers and interfaces for various scientific instruments.

AFL.automation.instrument

#### **AFL**.automation.instrument

#### **Modules**

**DummySAS** 

FileCamera

I22SAXS

NetworkCamera

SeabreezeUVVis

SpecScreen\_Driver

# AFL.automation.instrument.DummySAS

#### **Classes**

Driver(name[, defaults, overrides])

DummySAS([overrides])

# AFL.automation.instrument.DummySAS.Driver

 $\textbf{class} \ \texttt{AFL}. a \textbf{utomation.instrument.DummySAS.Driver} (\textit{name}, \textit{defaults=None}, \textit{overrides=None})$ 

Bases: object

**\_\_init\_\_**(name, defaults=None, overrides=None)

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.instrument.DummySAS.DummySAS

```
class AFL.automation.instrument.DummySAS.DummySAS(overrides=None)
    Bases: Driver
    __init__(overrides=None)
        connect to spec
```

# **Methods**

init([overrides])	connect to spec
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	
expose([name, exposure, nexp, block,])	
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object
set_object([serialized])	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

#### **Attributes**

```
defaults
```

```
defaults = {}
__init__(overrides=None)
     connect to spec
expose(name=None, exposure=None, nexp=1, block=True, reduce_data=True, measure_transmission=True,
        save_nexus=True)
status()
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
```

```
retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                  uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
class AFL.automation.instrument.DummySAS.DummySAS(overrides=None)
     defaults = {}
     __init__(overrides=None)
          connect to spec
     expose(name=None, exposure=None, nexp=1, block=True, reduce_data=True, measure_transmission=True,
             save_nexus=True)
     status()
```

## AFL.automation.instrument.FileCamera

## Classes

FileCamera()

## AFL.automation.instrument.FileCamera.FileCamera

```
class AFL.automation.instrument.FileCamera.FileCamera
Bases: object
__init__()
```

```
__init__()
collect(fname)

__init__()
collect(fname)

class AFL.automation.instrument.FileCamera.FileCamera
__init__()
collect(fname)
```

#### AFL.automation.instrument.I22SAXS

#### **Classes**

```
Driver(name[, defaults, overrides])

I22SAXS([overrides])
```

## AFL.automation.instrument.I22SAXS.Driver

```
class AFL.automation.instrument.I22SAXS.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.instrument.I22SAXS.I22SAXS

```
class AFL.automation.instrument.I22SAXS.I22SAXS(overrides=None, **kwargs)
    Bases: Driver
    __init__(overrides=None, **kwargs)
```

# **Methods**

init([overrides])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
	Store an object in the dropoox
execute(**kwargs)	
expose(name[, empty, nframes, acq_time,])	Perform a sequence of exposures at positions defined in self.config['pos_list'].
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
post_execute(**kwargs)	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
read_integrated(scanid)	Scans the appropriate directory for the filename of the data collected with filename
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
<pre>set_config(**kwargs)</pre>	
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	•
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

#### **Attributes**

```
defaults
```

```
defaults = {'acq_time': 1, 'acq_timeout': 120, 'address': '',
'data_read_cooldown': 5, 'empty_scan_id': '', 'file_read_timeout':
'nframes': 1, 'port': 2222, 'pos_list': [], 'processed_base_path':
'/mnt/i22_processed/i22-', 'reduced_data_suffix':
'_saxs_Transmission_Averaged_Subtracted_IvsQ_processed.nxs', 'ssh_key_path': '',
'username': ''}
__init__(overrides=None, **kwargs)
expose(name, empty=False, nframes=None, acq_time=None, set_empty=False)
    Perform a sequence of exposures at positions defined in self.config['pos_list'].
    Return the integrated data of the measurement with the lowest measured sample transmission.
read_integrated(scanid)
    Scans the appropriate directory for the filename of the data collected with filename
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
            • obj (object) – The object to store in the dropbox
            • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
    Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
```

Executed after each call to execute

All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden by subclasses.

```
pre_execute(**kwargs)
```

post\_execute(\*\*kwargs)

get\_sample()

Executed before each call to execute

All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden by subclasses.

queued()

```
quickbar()
     reset_sample()
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                 uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     status()
     unqueued()
class AFL.automation.instrument.I22SAXS.I22SAXS(overrides=None, **kwargs)
     defaults = {'acq_time': 1, 'acq_timeout': 120, 'address': '',
     'data_read_cooldown': 5, 'empty_scan_id': '', 'file_read_timeout': 30,
     'nframes': 1, 'port': 2222, 'pos_list': [], 'processed_base_path':
     '/mnt/i22_processed/i22-', 'reduced_data_suffix':
     '_saxs_Transmission_Averaged_Subtracted_IvsQ_processed.nxs', 'ssh_key_path': '',
     'username': ''}
     __init__(overrides=None, **kwargs)
     expose(name, empty=False, nframes=None, acq_time=None, set_empty=False)
          Perform a sequence of exposures at positions defined in self.config['pos_list'].
          Return the integrated data of the measurement with the lowest measured sample transmission.
     read_integrated(scanid)
          Scans the appropriate directory for the filename of the data collected with filename
```

#### AFL.automation.instrument.NetworkCamera

#### **Classes**

NetworkCamera(url)

# AFL.automation.instrument.NetworkCamera.NetworkCamera

```
class AFL.automation.instrument.NetworkCamera.NetworkCamera(url)
    Bases: object
    __init__(url)
```

#### **Methods**

```
__init__(url)

camera_reset()

collect()

__init__(url)
```

```
camera_reset()
collect()
class AFL.automation.instrument.NetworkCamera.NetworkCamera(url)
    __init__(url)
    camera_reset()
```

## AFL.automation.instrument.SeabreezeUVVis

## **Classes**

collect()

Driver(name[, defaults, overrides])	
Eq(key, value)	Query equality of a given key's value to the specified value.
Path(*args, **kwargs)	PurePath subclass that can make system calls.
SeabreezeUVVis([backend, device_serial,])	

# AFL.automation.instrument.SeabreezeUVVis.Driver

class AFL.automation.instrument.SeabreezeUVVis.Driver(name, defaults=None, overrides=None)
 Bases: object
 \_\_init\_\_(name, defaults=None, overrides=None)

## Methods

init(name[, defaults, overrides])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	·
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

unqueued()
queued()

quickbar()

```
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.instrument.SeabreezeUVVis.Eq

```
class AFL.automation.instrument.SeabreezeUVVis.Eq(key: str, value: Any)
```

Bases: NoBool

Query equality of a given key's value to the specified value.

See *Key* in this module for a more intuitive interface for equality.

#### **Parameters**

- **key** (*str*) e.g. "color", "sample.name"
- value (JSONSerializable) May be a string, number, list, or dict.

## **Examples**

Search for color == "red"

```
>>> c.search(Eq("color", "red"))
```

```
__init__(key: str, value: Any) \rightarrow None
```

# **Methods**

```
__init__(key, value)

decode(*, key, value)

encode()
```

## **Attributes**

```
key value
```

```
key: str
value: Any
encode()
classmethod decode(*, key, value)
__init__(key: str, value: Any) → None
```

## AFL.automation.instrument.SeabreezeUVVis.Path

class AFL.automation.instrument.SeabreezeUVVis.Path(\*args, \*\*kwargs)

Bases: PurePath

PurePath subclass that can make system calls.

Path represents a filesystem path but unlike PurePath, also offers methods to do system calls on path objects. Depending on your system, instantiating a Path will return either a PosixPath or a WindowsPath object. You can also instantiate a PosixPath or WindowsPath directly, but cannot instantiate a WindowsPath on a POSIX system or vice versa.

\_\_init\_\_()

#### **Methods**

init()  absolute()  Return an absolute version of this path by prepending the current working directory.  as_posix()  Return the string representation of the path with forward (/) slashes.  as_uri()  chmod(mode, *[, follow_symlinks])  chamod()  Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists()  expanduser()  Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists()  expanduser()  Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)  glob(pattern)  literate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group()  Return the group name of the file gid.  Make this path a hard link pointing to the same file as target.  home()  Return a new path pointing to the user's home directory (as returned by os.path.expanduser(~')).  is_absolute()  is_absolute()  is_absolute()  is_block_device()  is_char_device()  whether this path is a block device.  is_char_device()  whether this path is a directory.  is_fife()  whether this path is a directory.  is_fife()  whether this path is a regular file (also True for symlinks pointing to regular files).  check if this path is a POSIX mount point  return True if the path is relative to another path or False.  is_reserved()  Return True if the path is a symbolic link.  continues on pext nage.		
the current working directory.  Return the string representation of the path with forward (/) slashes.  as_uri() Return the path as a 'file' URI.  chmod(mode, *[, follow_symlinks]) Change the permissions of the path, like os.chmod().  cwd() Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists() Whether this path exists.  expanduser() Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)  Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group() Return the group name of the file gid.  Make this path a hard link pointing to the same file as target.  home() Return a new path pointing to the user's home directory (as returned by os.path.expanduser('~')).  is_absolute() True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device() Whether this path is a block device.  is_char_device() Whether this path is a directory.  is_fifo() Whether this path is a fIFO.  is_file() Whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount() Check if this path is a POSIX mount point  is_relative_to(*other) Return True if the path contains one of the special names reserved by the system, if any.  is_socket() Whether this path is a symbolic link.	init()	
ward (/) slashes.  as_uri() chmod(mode, *[, follow_symlinks]) Change the permissions of the path, like os.chmod().  Cwd() Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists() Whether this path exists.  expanduser() Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)  glob(pattern)  lterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group() Return the group name of the file gid.  hardlink_to(target) Make this path a hard link pointing to the same file as target.  home() Return a new path pointing to the user's home directory (as returned by os.path.expanduser(^-')).  is_absolute() True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device() Whether this path is a block device.  is_char_device() Whether this path is a character device.  is_dir() Whether this path is a firectory.  is_fifo() Whether this path is a regular file (also True for symllinks pointing to regular files).  is_mount() Check if this path is a POSIX mount point  relative_to(*other) Return True if the path is relative to another path or False.  is_reserved() Return True if the path is relative to another path or False.  is_socket() Whether this path is a socket.  is_symlink() Whether this path is a symbolic link.	absolute()	
chmod(mode, *[, follow_symlinks])       Change the permissions of the path, like os.chmod().         cwd()       Return a new path pointing to the current working directory (as returned by os.getcwd()).         exists()       Whether this path exists.         expanduser()       Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)         glob(pattern)       Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.         group()       Return the group name of the file gid.         hardlink_to(target)       Make this path a hard link pointing to the same file as target.         home()       Return a new path pointing to the user's home directory (as returned by os.path.expanduser('-')).         is_absolute()       True if the path is absolute (has both a root and, if applicable, a drive).         is_block_device()       Whether this path is a block device.         is_char_device()       Whether this path is a character device.         is_dir()       Whether this path is a directory.         is_file()       Whether this path is a regular file (also True for symlinks pointing to regular files).         is_mount()       Check if this path is a POSIX mount point         is_reserved()       Return True if the path contains one of the special names reserved by the system, if any.         is_socket()       Whether this path is a symbolic link. <td>as_posix()</td> <td>• •</td>	as_posix()	• •
cwd()  Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists()  Whether this path exists.  expanduser()  Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)  glob(pattern)  Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group()  Return the group name of the file gid.  Make this path a hard link pointing to the same file as target.  home()  Return a new path pointing to the user's home directory (as returned by os.path.expanduser(`~')).  is_absolute()  True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device()  is_char_device()  whether this path is a character device.  is_cis_dir()  Whether this path is a directory.  is_fifo()  whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount()  is_relative_to(*other)  Return True if the path is relative to another path or False.  is_reserved()  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  is_symlink()  Whether this path is a socket.	as_uri()	Return the path as a 'file' URI.
cwd()  Return a new path pointing to the current working directory (as returned by os.getcwd()).  exists()  Whether this path exists.  expanduser()  Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)  glob(pattern)  Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group()  Return the group name of the file gid.  Make this path a hard link pointing to the same file as target.  home()  Return a new path pointing to the user's home directory (as returned by os.path.expanduser(`~')).  is_absolute()  True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device()  is_char_device()  whether this path is a character device.  is_cis_dir()  Whether this path is a directory.  is_fifo()  whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount()  is_relative_to(*other)  Return True if the path is relative to another path or False.  is_reserved()  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  is_symlink()  Whether this path is a socket.	<pre>chmod(mode, *[, follow_symlinks])</pre>	Change the permissions of the path, like os.chmod().
expanduser()Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)glob(pattern)Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.group()Return the group name of the file gid.hardlink_to(target)Make this path a hard link pointing to the same file as target.home()Return a new path pointing to the user's home directory (as returned by os.path.expanduser('~')).is_absolute()True if the path is absolute (has both a root and, if applicable, a drive).is_block_device()Whether this path is a block device.is_char_device()Whether this path is a character device.is_dir()Whether this path is a directory.is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a symbolic link.	cwd()	Return a new path pointing to the current working di-
structs (as returned by os.path.expanduser)  glob(pattern)  Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.  group()  Return the group name of the file gid.  Make this path a hard link pointing to the same file as target.  home()  Return a new path pointing to the user's home directory (as returned by os.path.expanduser("~")).  is_absolute()  True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device()  whether this path is a block device.  is_char_device()  whether this path is a character device.  is_dir()  whether this path is a directory.  is_fifo()  whether this path is a FIFO.  is_file()  Whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount()  is_mount()  Check if this path is a POSIX mount point  is_relative_to(*other)  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  is_symlink()  Whether this path is a symbolic link.	exists()	Whether this path exists.
any kind, including directories) matching the given relative pattern.  group() Return the group name of the file gid.  hardlink_to(target) Make this path a hard link pointing to the same file as target.  home() Return a new path pointing to the user's home directory (as returned by os.path.expanduser('~')).  is_absolute() True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device() Whether this path is a block device. is_char_device() Whether this path is a character device. is_dir() Whether this path is a directory. is_fifo() Whether this path is a FIFO. is_file() Whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount() Check if this path is a POSIX mount point is_relative_to(*other) Return True if the path is relative to another path or False.  is_reserved() Return True if the path contains one of the special names reserved by the system, if any. is_socket() is_symlink() Whether this path is a symbolic link.	expanduser()	
hardlink_to(target)Make this path a hard link pointing to the same file as target.home()Return a new path pointing to the user's home direc- tory (as returned by os.path.expanduser('~')).is_absolute()True if the path is absolute (has both a root and, if applicable, a drive).is_block_device()Whether this path is a block device.is_char_device()Whether this path is a character device.is_dir()Whether this path is a directory.is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for sym- links pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a symbolic link.	glob(pattern)	any kind, including directories) matching the given
home()Return a new path pointing to the user's home directory (as returned by os.path.expanduser('~')).is_absolute()True if the path is absolute (has both a root and, if applicable, a drive).is_block_device()Whether this path is a block device.is_char_device()Whether this path is a character device.is_dir()Whether this path is a directory.is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a symbolic link.	group()	Return the group name of the file gid.
tory (as returned by os.path.expanduser('~')).  is_absolute()  True if the path is absolute (has both a root and, if applicable, a drive).  is_block_device()  Whether this path is a block device.  is_char_device()  Whether this path is a character device.  is_dir()  Whether this path is a directory.  is_fifo()  Whether this path is a FIFO.  is_file()  Whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount()  is_relative_to(*other)  Return True if the path is relative to another path or False.  is_reserved()  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  Whether this path is a symbolic link.	hardlink_to(target)	
applicable, a drive).  is_block_device()  Whether this path is a block device.  is_char_device()  Whether this path is a character device.  is_dir()  Whether this path is a directory.  is_fifo()  Whether this path is a FIFO.  is_file()  Whether this path is a regular file (also True for symlinks pointing to regular files).  is_mount()  is_relative_to(*other)  Return True if the path is relative to another path or False.  is_reserved()  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  is_symlink()  Whether this path is a symbolic link.	home()	
is_char_device()Whether this path is a character device.is_dir()Whether this path is a directory.is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_absolute()	*
is_dir()Whether this path is a directory.is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_block_device()	Whether this path is a block device.
is_fifo()Whether this path is a FIFO.is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_char_device()	Whether this path is a character device.
is_file()Whether this path is a regular file (also True for symlinks pointing to regular files).is_mount()Check if this path is a POSIX mount pointis_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_dir()	Whether this path is a directory.
links pointing to regular files).  is_mount() Check if this path is a POSIX mount point  is_relative_to(*other) Return True if the path is relative to another path or False.  is_reserved() Return True if the path contains one of the special names reserved by the system, if any.  is_socket() Whether this path is a socket.  is_symlink() Whether this path is a symbolic link.	is_fifo()	Whether this path is a FIFO.
is_relative_to(*other)Return True if the path is relative to another path or False.is_reserved()Return True if the path contains one of the special names reserved by the system, if any.is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_file()	
False.  is_reserved()  Return True if the path contains one of the special names reserved by the system, if any.  is_socket()  is_symlink()  Whether this path is a socket.  Whether this path is a symbolic link.	<pre>is_mount()</pre>	Check if this path is a POSIX mount point
is_socket()whether this path is a socket.is_symlink()Whether this path is a symbolic link.	<pre>is_relative_to(*other)</pre>	-
is_socket()Whether this path is a socket.is_symlink()Whether this path is a symbolic link.	is_reserved()	*
is_symlink() Whether this path is a symbolic link.	is_socket()	
	is_symlink()	Whether this path is a symbolic link.

continues on next page

Table 7 – continued from previous page

	d from previous page
<pre>iterdir()</pre>	Iterate over the files in this directory.
<pre>joinpath(*args)</pre>	Combine this path with one or several arguments, and
	return a new path representing either a subpath (if all
	arguments are relative paths) or a totally different path
	(if one of the arguments is anchored).
1chmod(mode)	Like chmod(), except if the path points to a symlink,
	the symlink's permissions are changed, rather than its
	target's.
link_to(target)	Make the target path a hard link pointing to this path.
lstat()	Like stat(), except if the path points to a symlink, the
V	symlink's status information is returned, rather than
	its target's.
match(path_pattern)	Return True if this path matches the given pattern.
mkdir([mode, parents, exist_ok])	Create a new directory at this given path.
open([mode, buffering, encoding, errors,])	Open the file pointed by this path and return a file
(t :) : (g) : (g) : (g)	object, as the built-in open() function does.
owner()	Return the login name of the file owner.
read_bytes()	Open the file in bytes mode, read it, and close the file.
<pre>read_text([encoding, errors])</pre>	Open the file in text mode, read it, and close the file.
readlink()	Return the path to which the symbolic link points.
relative_to(*other)	Return the relative path to another path identified by
,	the passed arguments.
rename(target)	Rename this path to the target path.
replace(target)	Rename this path to the target path, overwriting if that
	path exists.
resolve([strict])	Make the path absolute, resolving all symlinks on the
	way and also normalizing it.
rglob(pattern)	Recursively yield all existing files (of any kind, in-
	cluding directories) matching the given relative pat-
	tern, anywhere in this subtree.
rmdir()	Remove this directory.
<pre>samefile(other_path)</pre>	Return whether other_path is the same or not as this
	file (as returned by os.path.samefile()).
<pre>stat(*[, follow_symlinks])</pre>	Return the result of the stat() system call on this path,
	like os.stat() does.
<pre>symlink_to(target[, target_is_directory])</pre>	Make this path a symlink pointing to the target path.
touch([mode, exist_ok])	Create this file with the given access mode, if it
	doesn't exist.
<pre>unlink([missing_ok])</pre>	Remove this file or link.
with_name(name)	Return a new path with the file name changed.
<pre>with_stem(stem)</pre>	Return a new path with the stem changed.
<pre>with_suffix(suffix)</pre>	Return a new path with the file suffix changed.
write_bytes(data)	Open the file in bytes mode, write to it, and close the
	file.
<pre>write_text(data[, encoding, errors, newline])</pre>	Open the file in text mode, write to it, and close the
	file.

#### **Attributes**

anchor	The concatenation of the drive and root, or ".
drive	The drive prefix (letter or UNC path), if any.
name	The final path component, if any.
parent	The logical parent of the path.
parents	A sequence of this path's logical parents.
parts	An object providing sequence-like access to the com-
	ponents in the filesystem path.
root	The root of the path, if any.
stem	The final path component, minus its last suffix.
suffix	The final component's last suffix, if any.
suffixes	A list of the final component's suffixes, if any.

#### classmethod cwd()

Return a new path pointing to the current working directory (as returned by os.getcwd()).

#### classmethod home()

Return a new path pointing to the user's home directory (as returned by os.path.expanduser('~')).

## samefile(other\_path)

Return whether other\_path is the same or not as this file (as returned by os.path.samefile()).

#### iterdir()

Iterate over the files in this directory. Does not yield any result for the special paths '.' and '..'.

## glob(pattern)

Iterate over this subtree and yield all existing files (of any kind, including directories) matching the given relative pattern.

## rglob(pattern)

Recursively yield all existing files (of any kind, including directories) matching the given relative pattern, anywhere in this subtree.

#### absolute()

Return an absolute version of this path by prepending the current working directory. No normalization or symlink resolution is performed.

Use resolve() to get the canonical path to a file.

#### resolve(strict=False)

Make the path absolute, resolving all symlinks on the way and also normalizing it.

#### stat(\*, follow\_symlinks=True)

Return the result of the stat() system call on this path, like os.stat() does.

## owner()

Return the login name of the file owner.

#### group()

Return the group name of the file gid.

### **open**(*mode='r'*, *buffering=-1*, *encoding=None*, *errors=None*, *newline=None*)

Open the file pointed by this path and return a file object, as the built-in open() function does.

#### read\_bytes()

Open the file in bytes mode, read it, and close the file.

### read\_text(encoding=None, errors=None)

Open the file in text mode, read it, and close the file.

#### write\_bytes(data)

Open the file in bytes mode, write to it, and close the file.

#### write\_text(data, encoding=None, errors=None, newline=None)

Open the file in text mode, write to it, and close the file.

### readlink()

Return the path to which the symbolic link points.

#### touch(mode=438, exist\_ok=True)

Create this file with the given access mode, if it doesn't exist.

#### **mkdir**(mode=511, parents=False, exist\_ok=False)

Create a new directory at this given path.

### chmod(mode, \*, follow\_symlinks=True)

Change the permissions of the path, like os.chmod().

## lchmod(mode)

Like chmod(), except if the path points to a symlink, the symlink's permissions are changed, rather than its target's.

## unlink(missing\_ok=False)

Remove this file or link. If the path is a directory, use rmdir() instead.

## rmdir()

Remove this directory. The directory must be empty.

#### lstat()

Like stat(), except if the path points to a symlink, the symlink's status information is returned, rather than its target's.

## rename(target)

Rename this path to the target path.

The target path may be absolute or relative. Relative paths are interpreted relative to the current working directory, *not* the directory of the Path object.

Returns the new Path instance pointing to the target path.

#### replace(target)

Rename this path to the target path, overwriting if that path exists.

The target path may be absolute or relative. Relative paths are interpreted relative to the current working directory, *not* the directory of the Path object.

Returns the new Path instance pointing to the target path.

## symlink\_to(target, target\_is\_directory=False)

Make this path a symlink pointing to the target path. Note the order of arguments (link, target) is the reverse of os.symlink.

# hardlink\_to(target)

Make this path a hard link pointing to the same file as target.

Note the order of arguments (self, target) is the reverse of os.link's.

### link\_to(target)

Make the target path a hard link pointing to this path.

Note this function does not make this path a hard link to *target*, despite the implication of the function and argument names. The order of arguments (target, link) is the reverse of Path.symlink\_to, but matches that of os.link.

Deprecated since Python 3.10 and scheduled for removal in Python 3.12. Use hardlink\_to() instead.

#### exists()

Whether this path exists.

#### is\_dir()

Whether this path is a directory.

#### is\_file()

Whether this path is a regular file (also True for symlinks pointing to regular files).

# is\_mount()

Check if this path is a POSIX mount point

#### is symlink()

Whether this path is a symbolic link.

### is\_block\_device()

Whether this path is a block device.

### is\_char\_device()

Whether this path is a character device.

#### is\_fifo()

Whether this path is a FIFO.

### is\_socket()

Whether this path is a socket.

#### expanduser()

Return a new path with expanded ~ and ~user constructs (as returned by os.path.expanduser)

#### \_\_bytes\_\_()

Return the bytes representation of the path. This is only recommended to use under Unix.

### \_\_str\_\_()

Return the string representation of the path, suitable for passing to system calls.

# property anchor

The concatenation of the drive and root, or ".

### as\_posix()

Return the string representation of the path with forward (/) slashes.

### as\_uri()

Return the path as a 'file' URI.

#### property drive

The drive prefix (letter or UNC path), if any.

### is\_absolute()

True if the path is absolute (has both a root and, if applicable, a drive).

### is\_relative\_to(\*other)

Return True if the path is relative to another path or False.

### is\_reserved()

Return True if the path contains one of the special names reserved by the system, if any.

# joinpath(\*args)

Combine this path with one or several arguments, and return a new path representing either a subpath (if all arguments are relative paths) or a totally different path (if one of the arguments is anchored).

### match(path\_pattern)

Return True if this path matches the given pattern.

#### property name

The final path component, if any.

# property parent

The logical parent of the path.

#### property parents

A sequence of this path's logical parents.

#### property parts

An object providing sequence-like access to the components in the filesystem path.

#### relative\_to(\*other)

Return the relative path to another path identified by the passed arguments. If the operation is not possible (because this is not a subpath of the other path), raise ValueError.

#### property root

The root of the path, if any.

### property stem

The final path component, minus its last suffix.

### property suffix

The final component's last suffix, if any.

This includes the leading period. For example: '.txt'

### property suffixes

A list of the final component's suffixes, if any.

These include the leading periods. For example: ['.tar', '.gz']

#### with\_name(name)

Return a new path with the file name changed.

#### with\_stem(stem)

Return a new path with the stem changed.

# with\_suffix(suffix)

Return a new path with the file suffix changed. If the path has no suffix, add given suffix. If the given suffix is an empty string, remove the suffix from the path.

#### AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis

Bases: Driver
\_\_init\_\_(backend='cseabreeze', device\_serial=None, overrides=None)

### **Methods**

```
__init__([backend, device_serial, overrides])
collect(nframes[, reduced, absorbance, ...])
collectContinuous(duration[, start, return_data])
collectSingleSpectrum([set_reference, set_air])
                                                   Store an object in the dropbox
deposit_obj(obj[, uid])
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
getExposure()
getExposureDelay()
getFilename()
getFilepath()
getSaveSingleScan()
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
                                                   Executed after each call to execute
post_execute(**kwargs)
pre_execute(**kwargs)
                                                   Executed before each call to execute
queued()
quickbar()
reduced(data_raw_mean, data_raw_std[, ...])
```

continues on next page

Table 8 – continued from previous page

```
reset_sample()

retrieve_obj(uid[, delete]) Retrieve an object from the dropbox

setExposure(time)

setExposureDelay(time)

setFilename(filename)

setFilepath(filepath)

setSaveSingleScan(saveSingleScan)

set_config(**kwargs)

set_data(data) Set data in the DataPacket object

set_object([serialized])

set_sample(sample_name[, sample_uuid])

status()

unqueued()
```

### **Attributes**

```
defaults

defaults = {'air_uuid': '', 'correctDarkCounts': False, 'correctNonlinearity':
False, 'exposure': 0.01, 'exposure_delay': 0, 'filename': 'test.h5', 'filepath':
'.', 'reference_uuid': '', 'saveSingleScan': False}
__init__(backend='cseabreeze', device_serial=None, overrides=None)
getExposure()
getExposureDelay()
getFilename()
getSaveSingleScan()
getFilepath()
setFilepath(filepath)
setFilename(filename)
setSaveSingleScan(saveSingleScan)
```

```
setExposureDelay(time)
setExposure(time)
collectContinuous(duration, start=None, return_data=False, **kwargs)
collectSingleSpectrum(set_reference=False, set_air=False, **kwargs)
collect(nframes: int, reduced: bool = False, absorbance: bool = True, set\_reference: bool = False, set\_air:
         bool = False, exposure: float | None = None, return_data: bool = False, **kwargs)
reduced(data_raw_mean, data_raw_std, absorbance=True, reference_uuid='reference_uuid')
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
```

```
set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     status()
     unqueued()
class AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis(backend='cseabreeze',
                                                                      device_serial=None,
                                                                      overrides=None)
     defaults = {'air_uuid': '', 'correctDarkCounts': False, 'correctNonlinearity':
     False, 'exposure': 0.01, 'exposure_delay': 0, 'filename': 'test.h5', 'filepath':
     '.', 'reference_uuid': '', 'saveSingleScan': False}
     __init__(backend='cseabreeze', device_serial=None, overrides=None)
     getExposure()
     getExposureDelay()
     getFilename()
     getSaveSingleScan()
     getFilepath()
     setFilepath(filepath)
     setFilename(filename)
     setSaveSingleScan(saveSingleScan)
     setExposureDelay(time)
     setExposure(time)
     collectContinuous(duration, start=None, return_data=False, **kwargs)
     collectSingleSpectrum(set_reference=False, set_air=False, **kwargs)
     collect(nframes: int, reduced: bool = False, absorbance: bool = True, set reference: bool = False, set air:
              bool = False, exposure: float | None = None, return_data: bool = False, **kwargs)
     reduced(data_raw_mean, data_raw_std, absorbance=True, reference_uuid='reference_uuid')
```

# AFL.automation.instrument.SpecScreen\_Driver

#### **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x, l)	Return the square root of x.

## AFL.automation.instrument.SpecScreen Driver.ceil

```
AFL.automation.instrument.SpecScreen_Driver.ceil(x,/)
Return the ceiling of x as an Integral.

This is the smallest integer >= x.
```

### AFL.automation.instrument.SpecScreen\_Driver.listify

```
{\tt AFL.automation.instrument.SpecScreen\_Driver. \textbf{listify}}(obj)
```

### AFL.automation.instrument.SpecScreen Driver.sqrt

```
AFL.automation.instrument.SpecScreen_Driver.\mathbf{sqrt}(x,/)
Return the square root of x.
```

### Classes

```
Driver(name[, defaults, overrides])
SpecScreen_Driver([log_file])
```

### AFL.automation.instrument.SpecScreen Driver.Driver

```
class AFL.automation.instrument.SpecScreen_Driver.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

### **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.instrument.SpecScreen\_Driver.SpecScreen\_Driver

```
class AFL.automation.instrument.SpecScreen_Driver.SpecScreen_Driver(log_file=None)
    Bases: Driver
    __init__(log_file=None)
```

#### **Methods**

```
_init__([log_file])
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
post_execute(**kwargs)
                                                   Executed after each call to execute
                                                   Executed before each call to execute
pre_execute(**kwargs)
queued()
quickbar()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                   Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
unqueued()
```

```
__init__(log_file=None)
status()
execute(**kwargs)
```

```
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
unqueued()
```

```
class AFL.automation.instrument.SpecScreen_Driver.SpecScreen_Driver(log_file=None)
    __init__(log_file=None)
    status()
    execute(**kwargs)
```

# 6.2.3 Loading

The Loading module provides functionality for loading and handling samples.

```
AFL.automation.loading
```

# AFL.automation.loading

# Modules

CetoniMultiPosValve	
ChemyxSyringePump	This is largely duplicated from the reference code provided by Chemyx.
DigitalOutPressureController	race of Chemyn.
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
LoadStopperDriver	
MultiChannelRelay	
NE1kSyringePump	
OneSelectorBlowoutSampleCell	
PneumaticPressureSampleCell	
PneumaticSampleCell	
PressureController	
PressureControllerAsPump	
PushPullSelectorSampleCell	
RSoXSSolutionSampleCell	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
SyringePump	
Tubing	
TwoSelectorBlowoutSampleCell	
UltimusVPressureController	
- ViciMultiposSelector 6.2. Modules	225

### AFL.automation.loading.CetoniMultiPosValve

#### **Classes**

```
CetoniMultiPosValve(parentpump[, portlabels])

FlowSelector()
```

### AFL.automation.loading.CetoniMultiPosValve.CetoniMultiPosValve

Bases: FlowSelector
\_\_init\_\_(parentpump, portlabels={})
connect to valve and query the number of positions

### **Parameters**

- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

### **Methods**

init(parentpump[, portlabels])	connect to valve and query the number of positions
<pre>getPort([as_str])</pre>	query the current selected position
<pre>selectPort(port[, direction])</pre>	moves the selector to portnum

```
__init__(parentpump, portlabels={})
```

connect to valve and query the number of positions

#### **Parameters**

- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

```
selectPort(port, direction=None)
          moves the selector to portnum
          if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value,
          will move via most efficient route.
     getPort(as_str=False)
          query the current selected position
AFL.automation.loading.CetoniMultiPosValve.FlowSelector
class AFL.automation.loading.CetoniMultiPosValve.FlowSelector
     Bases: object
     __init__()
     Methods
        __init__()
       getPort()
       selectPort()
     getPort()
     selectPort()
class AFL.automation.loading.CetoniMultiPosValve.CetoniMultiPosValve(parentpump,
                                                                                 portlabels={})
     __init__(parentpump, portlabels={})
          connect to valve and query the number of positions
              Parameters
                                  (port - string describing the serial port the actuator is
                   • to
                    connected)
                   • use (baud - baudrate to)

    naming

                                    (portlabels - dict for smart port)
                                                                                             3,'instru-
                    ment':4,'rinse':5,'waste':6}
                   • {'sample' (of the form) - 3,'instrument':4,'rinse':5,'waste':6}
     selectPort(port, direction=None)
          moves the selector to portnum
          if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value,
          will move via most efficient route.
```

6.2. Modules 227

getPort(as\_str=False)

query the current selected position

# AFL.automation.loading.ChemyxSyringePump

This is largely duplicated from the reference code provided by Chemyx. Their package is a GUI and direct import of the module would be problematic.

### **Functions**

<pre>getOpenPorts()</pre>	
parsePortName(portinfo)	On macOS and Linux, selects only usbserial options and parses the 8 character serial number.

### AFL.automation.loading.ChemyxSyringePump.getOpenPorts

AFL.automation.loading.ChemyxSyringePump.getOpenPorts()

### AFL.automation.loading.ChemyxSyringePump.parsePortName

AFL.automation.loading.ChemyxSyringePump.parsePortName(portinfo)

On macOS and Linux, selects only usbserial options and parses the 8 character serial number.

### Classes

```
ChemyxConnection(port, baudrate[, x, mode, ...])

ChemyxSyringePump(port, syringe_id_mm, ...)

SyringePump()
```

### AFL.automation.loading.ChemyxSyringePump.ChemyxConnection

```
Bases: object
__init__(port, baudrate, x=0, mode=0, verbose=False)
```

### **Methods**

```
_init__(port, baudrate[, x, mode, verbose])
 addMode(command)
 addX(command)
 closeConnection()
 getDisplacedVolume()
 getElapsedTime()
 getParameterLimits()
 getParameters()
 getPumpStatus()
 getResponse()
 openConnection()
 pausePump()
 restartPump()
 sendCommand(command)
 setDelay(delay)
 setDiameter(diameter)
 setRate(rate)
 setTime(timer)
 setUnits(units)
 setVolume(volume)
 startPump()
 stopPump()
__init__(port, baudrate, x=0, mode=0, verbose=False)
openConnection()
closeConnection()
```

```
sendCommand(command)
getResponse()
startPump()
stopPump()
pausePump()
restartPump()
setUnits(units)
setDiameter(diameter)
setRate(rate)
setVolume(volume)
setDelay(delay)
setTime(timer)
getParameterLimits()
getParameters()
getDisplacedVolume()
getElapsedTime()
getPumpStatus()
addMode(command)
addX(command)
```

# AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump

### **Methods**

getLevel()

```
_init__(port, syringe_id_mm, syringe_volume)
                                                     Initializes and verifies connection to a Chemyx sy-
                                                     ringe pump.
 blockUntilStatusStopped([pollingdelay])
                                                     This is a deprecated function from old serial logic.
 dispense(volume[, block, delay])
 emptySyringe()
 getLevel()
 getRate()
 getStatus()
                                                     query the pump status and return whether the pump
                                                     is moving or not (true if moving, false if stopped)
 getValueFromParams(search key)
 setLevel(level)
 setRate(rate)
                                                     Abort the current dispense/withdraw action.
 stop()
                                                     The function waits until the last dosage command has
 wait_dosage_finished([timeout_seconds])
                                                     finished until the timeout occurs.
 withdraw(volume[, block, delay])
__init__(port, syringe_id_mm, syringe_volume, baud=9600, flow_delay=5)
     Initializes and verifies connection to a Chemyx syringe pump.
         port = serial port reference
         syringe id mm = syringe inner diameter in mm, used for absolute volume.
             (will re-program the pump with this diameter on connection)
         syringe_volume = syringe volume in mL
         baud = baudrate for connection
wait_dosage_finished(timeout seconds=60)
     The function waits until the last dosage command has finished until the timeout occurs.
stop()
     Abort the current dispense/withdraw action.
withdraw(volume, block=True, delay=True)
dispense(volume, block=True, delay=True)
setRate(rate)
getRate()
emptySyringe()
```

```
blockUntilStatusStopped(pollingdelay=0.2)
    This is a deprecated function from old serial logic. It should work, but do not use.
getStatus()
    query the pump status and return whether the pump is moving or not (true if moving, false if stopped)
getValueFromParams(search_key)
```

### AFL.automation.loading.ChemyxSyringePump.SyringePump

```
class AFL.automation.loading.ChemyxSyringePump.SyringePump
    Bases: object
    __init__()
```

#### **Methods**

```
__init__()
dispense(volume[, block])
emptySyringe()
getRate(rate)
setRate(rate)
stop()
withdraw(volume[, block])
```

```
stop()
withdraw(volume, block=True)
dispense(volume, block=True)
setRate(rate)
getRate(rate)
emptySyringe()
```

```
__init__(port, syringe_id_mm, syringe_volume, baud=9600, flow_delay=5)
          Initializes and verifies connection to a Chemyx syringe pump.
              port = serial port reference
              syringe id mm = syringe inner diameter in mm, used for absolute volume.
                  (will re-program the pump with this diameter on connection)
              syringe_volume = syringe volume in mL
              baud = baudrate for connection
     wait_dosage_finished(timeout_seconds=60)
          The function waits until the last dosage command has finished until the timeout occurs.
     stop()
          Abort the current dispense/withdraw action.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     getLevel()
     setLevel(level)
     blockUntilStatusStopped(pollingdelay=0.2)
          This is a deprecated function from old serial logic. It should work, but do not use.
     getStatus()
          query the pump status and return whether the pump is moving or not (true if moving, false if stopped)
     getValueFromParams(search_key)
AFL.automation.loading.ChemyxSyringePump.getOpenPorts()
AFL.automation.loading.ChemyxSyringePump.parsePortName(portinfo)
     On macOS and Linux, selects only usbserial options and parses the 8 character serial number.
class AFL.automation.loading.ChemyxSyringePump.ChemyxConnection(port, baudrate, x=0, mode=0,
                                                                           verbose=False)
     __init__(port, baudrate, x=0, mode=0, verbose=False)
     openConnection()
     closeConnection()
     sendCommand(command)
     getResponse()
     startPump()
     stopPump()
```

```
pausePump()
restartPump()
setUnits(units)
setDiameter(diameter)
setRate(rate)
setVolume(volume)
setDelay(delay)
setTime(timer)
getParameterLimits()
getParameters()
getDisplacedVolume()
getElapsedTime()
getPumpStatus()
addMode(command)
addX(command)
```

# AFL.automation.loading.DigitalOutPressureController

### **Classes**

234

```
DigitalOutPressureController(digital_out, ...)

PressureController()

Abstract superclass for pressure controllers that provides timed dispensing
```

# AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController

```
 \textbf{class} \  \, \textbf{AFL}. \textbf{automation.loading.DigitalOutPressureController.DigitalOutPressureController} ( \textit{digital\_out}, \\ \textit{pres-sure\_to\_v\_conv})
```

```
Bases: PressureController
__init__(digital_out, pressure_to_v_conv)
Initializes a DigitalOutPressureController
Params:
digital_out (AFL.automation.DigitalOut): pressure_to_v_conv (float): pressure units per volt
```

### **Methods**

init(digital_out, pressure_to_v_conv)	Initializes a DigitalOutPressureController
<pre>blockUntilStatusStopped([pollingdelay])</pre>	block execution until the controller finishes a dispense
dispenseRunning()	Returns true if a timed dispense is running, false otherwise.
<pre>ramp_dispense(dispense_start_pressure,)</pre>	Perform a pressure dispense with a linear ramp in pressure between <i>dispense_start_pressure</i> and <i>dispense_stop_pressure</i> , stopping after <i>dispense_time</i> .
set_P(pressure)	
stop()	Abort the current timed dispense action.
<pre>timed_dispense(dispense_pressure,</pre>	s- Perform a pressure dispense at pressure dispense_pressure, stopping after dispense_time.

**\_\_init\_\_**(*digital\_out*, *pressure\_to\_v\_conv*)

Initializes a DigitalOutPressureController

Params:

digital\_out (AFL.automation.DigitalOut): pressure\_to\_v\_conv (float): pressure units per volt

set\_P(pressure)

## blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

### dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

### stop()

Abort the current timed dispense action.

### timed\_dispense(dispense\_pressure, dispense\_time, block=True)

Perform a pressure dispense at pressure dispense\_pressure, stopping after dispense\_time. This dispense can be interrupted by calling self.stop().

### AFL.automation.loading.DigitalOutPressureController.PressureController

### ${\bf class} \ \, {\tt AFL.automation.loading.DigitalOutPressureController.} \\ {\bf PressureController}. \\ {\bf class} \ \, {\tt AFL.automation.loading.DigitalOutPressureController.} \\ {\bf class \ \, {\tt AFL.aut$

Bases: object

Abstract superclass for pressure controllers that provides timed dispensing

\_\_init\_\_()

### **Methods**

init()	
<pre>blockUntilStatusStopped([pollingdelay])</pre>	block execution until the controller finishes a dispense
dispenseRunning()	Returns true if a timed dispense is running, false otherwise.
<pre>ramp_dispense(dispense_start_pressure,)</pre>	Perform a pressure dispense with a linear ramp in pressure between <i>dispense_start_pressure</i> and <i>dispense_stop_pressure</i> , stopping after <i>dispense_time</i> .
stop()	Abort the current timed dispense action.
<pre>timed_dispense(dispense_pressure, pense_time)</pre>	Perform a pressure dispense at pressure dispense_pressure, stopping after dispense_time.

# ${\tt timed\_dispense}(\textit{dispense\_pressure}, \textit{dispense\_time}, \textit{block=True})$

Perform a pressure dispense at pressure *dispense\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop().

### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

# dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

### stop()

Abort the current timed dispense action.

 $\textbf{class} \ \texttt{AFL}. automation. loading. \texttt{DigitalOutPressureController}. \textbf{\textit{DigitalOutPressureController}(} \textit{\textit{digital\_out}}, \\$ 

pressure\_to\_v\_conv)

```
__init__(digital_out, pressure_to_v_conv)
```

Initializes a DigitalOutPressureController

Params:

digital\_out (AFL.automation.DigitalOut): pressure\_to\_v\_conv (float): pressure units per volt

```
set_P(pressure)
```

### AFL.automation.loading.DoubleViciMultiposSelector

#### **Classes**

```
DoubleViciMultiposSelector(port1, port2[, ...])

FlowSelector()

ViciMultiposSelector(port[, baudrate, ...])
```

# AFL.automation.loading.DoubleViciMultiposSelector.DoubleViciMultiposSelector

```
class AFL.automation.loading.DoubleViciMultiposSelector.DoubleViciMultiposSelector(port1,
```

port2, baudrate=9600, portlabels=None)

```
Bases: FlowSelector
```

```
__init__(port1, port2, baudrate=9600, portlabels=None) connect to valve and query the number of positions
```

### **Parameters**

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

### **Methods**

init(port1, port2[, baudrate, portlabels])	connect to valve and query the number of positions
<pre>getPort([as_str])</pre>	query the current selected position
<pre>selectPort(port[, direction])</pre>	moves the selector to portnum

```
__init__(port1, port2, baudrate=9600, portlabels=None) connect to valve and query the number of positions
```

#### **Parameters**

```
use (baud - baudrate to)
naming (portlabels - dict for smart port) - 3,'instrument':4,'rinse':5,'waste':6}
{'sample' (of the form) - 3,'instrument':4,'rinse':5,'waste':6}
selectPort(port, direction=None)
moves the selector to portnum
if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.
getPort(as_str=False)
query the current selected position
```

### AFL.automation.loading.DoubleViciMultiposSelector.FlowSelector

```
class AFL.automation.loading.DoubleViciMultiposSelector.FlowSelector
    Bases: object
    __init__()
```

### **Methods**

```
__init__()

getPort()

selectPort()
```

getPort()
selectPort()

# AFL.automation.loading.DoubleViciMultiposSelector.ViciMultiposSelector

```
Bases: SerialDevice, FlowSelector
__init__(port, baudrate=9600, portlabels=None)
connect to valve and query the number of positions
```

#### **Parameters**

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)

```
• naming (portlabels - dict for smart port) - 3,'instrument':4,'rinse':5,'waste':6}
```

• {'sample' (of the form) - 3,'instrument':4,'rinse':5,'waste':6}

#### **Methods**

init(port[, baudrate, portlabels])	connect to valve and query the number of positions
<pre>getPort([as_str])</pre>	query the current selected position
<pre>selectPort(port[, direction, block])</pre>	moves the selector to portnum
<pre>sendCommand(cmd[, response, questionmarkOK,])</pre>	

\_\_init\_\_(port, baudrate=9600, portlabels=None)

connect to valve and query the number of positions

#### **Parameters**

- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

selectPort(port, direction=None, block=True)

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

#### **Parameters**

- port (String or int) the name, or number, of the port to switch to
- **direction** (*String*, *default=None*) direction "CW" or "CCW" to perform the move in
- **block** (bool, default=True) block return until move completes

```
getPort(as_str=False)
```

query the current selected position

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

 $\textbf{class} \ \texttt{AFL}. automation. loading. Double \texttt{ViciMultiposSelector}. \textbf{\textit{DoubleViciMultiposSelector}(port1, port1) and \texttt{\textit{Notion}(port1)} and \texttt{$ 

port2, baudrate=9600, portlabels=None)

\_\_init\_\_(port1, port2, baudrate=9600, portlabels=None) connect to valve and query the number of positions

#### **Parameters**

```
• use (baud - baudrate to)
```

```
• naming (portlabels - dict for smart port) - 3,'instrument':4,'rinse':5,'waste':6}
```

```
• {'sample' (of the form) - 3,'instrument':4,'rinse':5,'waste':6}
```

```
selectPort(port, direction=None)
```

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

```
getPort(as_str=False)
```

query the current selected position

# AFL.automation.loading.DummyPump

#### **Classes**

```
DummyPump()

SerialDevice(port[, baudrate, timeout, ...])

SyringePump()
```

### AFL.automation.loading.DummyPump.DummyPump

```
class AFL.automation.loading.DummyPump.DummyPump
     Bases: SyringePump
    __init__()
```

Dummy pump for testing - does nothing, but boy does it look good doing it.

### **Methods**

```
__init__()
```

Dummy pump for testing - does nothing, but boy does it look good doing it.

### stop()

Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.

```
withdraw(volume, block=True, delay=True)
```

dispense(volume, block=True, delay=True)

setRate(rate)

getRate()

emptySyringe()

blockUntilStatusStopped(pollingdelay=0.2)

### getStatus()

query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)

# AFL.automation.loading.DummyPump.SerialDevice

```
\begin{tabular}{ll} \textbf{class} & \textbf{AFL}. \textbf{automation.loading.DummyPump.SerialDevice}(port, baudrate=19200, timeout=0.5, \\ & raw\_writes=False) \end{tabular} Bases: object
```

\_\_init\_\_(port, baudrate=19200, timeout=0.5, raw\_writes=False)

### **Methods**

```
__init__(port[, baudrate, timeout, raw_writes])
 sendCommand(cmd[, response, questionmarkOK,
 ...])
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

**sendCommand**(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)

# AFL.automation.loading.DummyPump.SyringePump

```
class AFL.automation.loading.DummyPump.SyringePump
     Bases: object
     __init__()
```

### **Methods**

```
__init__()
dispense(volume[, block])
emptySyringe()
getRate(rate)
setRate(rate)
stop()
withdraw(volume[, block])
```

```
stop()
     withdraw(volume, block=True)
     dispense(volume, block=True)
     setRate(rate)
     getRate(rate)
     emptySyringe()
class AFL.automation.loading.DummyPump.DummyPump
     __init__()
```

Dummy pump for testing - does nothing, but boy does it look good doing it.

```
stop()
          Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     blockUntilStatusStopped(pollingdelay=0.2)
     getStatus()
          query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
AFL.automation.loading.FlowSelector
Classes
 FlowSelector()
AFL.automation.loading.FlowSelector.FlowSelector
class AFL.automation.loading.FlowSelector.FlowSelector
     Bases: object
     __init__()
     Methods
      __init__()
      getPort()
      selectPort()
     getPort()
     selectPort()
class AFL.automation.loading.FlowSelector.FlowSelector
     getPort()
     selectPort()
```

# AFL.automation.loading.LoadStopperDriver

#### Classes

<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
<pre>Driver(name[, defaults, overrides])</pre>	
<pre>LoadStopperDriver(sensor[, load_client,])</pre>	Driver for stopping loads
SensorPollingThread(sensor[, period,])	
StopLoadCBv1(poll, period, load_client[,])	
StopLoadCBv2(poll, period[, load_client,])	

# AFL.automation.loading.LoadStopperDriver.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

**\_\_init\_\_**(*ip=None*, *port='5000'*, *username=None*, *interactive=False*)

### **Methods**

init([ip, port, username, interactive])	
<pre>clear_history()</pre>	
<pre>clear_queue()</pre>	
debug(state)	
<pre>deposit_obj(obj[, uid])</pre>	Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object under if not specified, a new uuid will be generated
<pre>driver_status()</pre>	
enqueue([interactive])	
enqueued_base(**kwargs)	
<pre>from_server_name(server_name, **kwargs)</pre>	

continues on next page

### Table 9 – continued from previous page

```
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
get_unqueued_commands([inherit_commands])
halt()
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
remove_item(uuid)
reset_queue_daemon()
                                                  Retrieve an object from the dropbox id: str, the uuid
retrieve_obj(uid[, delete])
                                                  of the object to retrieve delete: bool, if True, delete
                                                  the object after retrieving
server_cmd(cmd, **kwargs)
set_config([interactive])
set_driver_object(**kw)
set_object([serialize])
unqueued_base(**kwargs)
wait([target_uuid, interval, for_history, ...])
```

\_\_init\_\_(ip=None, port='5000', username=None, interactive=False)

classmethod from\_server\_name(server\_name, \*\*kwargs)

```
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
set_driver_object(**kw)
get_driver_object(name)
deposit_obj(obj, uid=None)
     Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
     under if not specified, a new uuid will be generated
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
     the object after retrieving
```

```
set_object(serialize=True, **kw)
get_object(name, serialize=True)
```

### AFL.automation.loading.LoadStopperDriver.Driver

```
class AFL.automation.loading.LoadStopperDriver.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

#### **Methods**

```
__init__(name[, defaults, overrides])
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
post_execute(**kwargs)
                                                   Executed after each call to execute
pre_execute(**kwargs)
                                                   Executed before each call to execute
queued()
quickbar()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                   Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
unqueued()
```

### unqueued()

```
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
             • data (dict) – Dictionary of data to store in the driver object
             • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
```

deposit\_obj(obj, uid=None)

Store an object in the dropbox

#### **Parameters**

- **obj** (*object*) The object to store in the dropbox
- **uid** (*str*) The uuid to store the object under

# AFL.automation.loading.LoadStopperDriver.LoadStopperDriver

 $\textbf{class} \ AFL. automation. loading. Load Stopper Driver. \textbf{\textit{Load Stopper Driver}} (sensor, load\_client = None, load\_client) and the load of the lo$ 

load\_object=None, auto\_initialize=True, overrides=None, data=None, sensorlabel=", name='LoadStopperDriver')

Bases: Driver

Driver for stopping loads

\_\_init\_\_(sensor, load\_client=None, load\_object=None, auto\_initialize=True, overrides=None, data=None, sensorlabel=", name='LoadStopperDriver')

#### **Methods**

```
_init__(sensor[, load_client, load_object, ...])
calibrate_sensor()
                                                    Store an object in the dropbox
deposit_obj(obj[, uid])
execute(**kwargs)
gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
{\tt get\_config}(name[,\,print\_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
post_execute(**kwargs)
                                                    Executed after each call to execute
pre_execute(**kwargs)
                                                    Executed before each call to execute
queued()
quickbar()
read_pol1()
read_poll_load()
read_sensor()
reset()
reset_poll()
reset_sample()
reset_stopper()
retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                    Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
unqueued()
```

#### **Attributes**

```
app
 defaults
defaults = {'load_speed': 2, 'period': 0.05, 'poll_window': 1000, 'sensorlabel':
'', 'stopper_baseline_duration': 2, 'stopper_filepath':
'/github/home/.afl/loadstopper_data', 'stopper_loadstop_cooldown': 2,
'stopper_min_load_time': 3, 'stopper_post_detection_sleep': 1,
'stopper_threshold_npts': 50, 'stopper_threshold_std': 3.0,
'stopper_threshold_v_step': 1, 'stopper_timeout': 120}
__init__(sensor, load_client=None, load_object=None, auto_initialize=True, overrides=None, data=None,
         sensorlabel=", name='LoadStopperDriver')
status()
property app
calibrate_sensor()
read_sensor()
read_poll()
read_poll_load()
reset()
reset_poll()
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
            • obj (object) – The object to store in the dropbox
            • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
    Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
```

```
post_execute(**kwargs)
          Executed after each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     pre_execute(**kwargs)
          Executed before each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     queued()
     quickbar()
     reset_sample()
     reset_stopper()
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                  uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                   • data (dict) – Dictionary of data to store in the driver object
                   • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
AFL.automation.loading.LoadStopperDriver.SensorPollingThread
class AFL.automation.loading.LoadStopperDriver.SensorPollingThread(sensor, period=0.1,
                                                                                callback=None,
                                                                                hv_pipe=None,
                                                                                window=None,
                                                                                filename=None,
                                                                                daemon=True, data=None)
     Bases: Thread
     __init__(sensor, period=0.1, callback=None, hv_pipe=None, window=None, filename=None,
                daemon=True, data=None)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

init(sensor[, period, callback,])	This constructor should always be called with keyword arguments.
alive()	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
read_load_buffer()	
reset_load_buffer()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(sensor, period=0.1, callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### read()

#### read\_load\_buffer()

reset\_load\_buffer()

#### terminate()

alive()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

# join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

# AFL.automation.loading.LoadStopperDriver.StopLoadCBv1

class AFL.automation.loading.LoadStopperDriver.StopLoadCBv1(poll, period, load\_client,

threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

Bases: SensorCallbackThread

\_\_init\_\_(poll, period, load\_client, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

init(poll, period, load_client[,])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal()</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(poll, period, load\_client, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### process\_signal()

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

# is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

# property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

## property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### terminate()

```
update_status(value)
```

#### AFL.automation.loading.LoadStopperDriver.StopLoadCBv2

```
class AFL.automation.loading.LoadStopperDriver.StopLoadCBv2(poll, period, load_client=None,
```

```
load_object=None,
threshold_npts=20,
threshold_v_step=1,
threshold_std=2.5, timeout=120,
min_load_time=3,
loadstop_cooldown=2,
post_detection_sleep=0.2,
baseline_duration=2,
trigger_on_end=False,
instatrigger=True, daemon=True,
filepath=None, data=None,
sensorlabel=")
```

#### Bases: SensorCallbackThread

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

init(poll, period[, load_client,])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal()</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### process\_signal()

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### terminate()

```
update_status(value)
```

'/github/home/.afl/loadstopper\_data', 'stopper\_loadstop\_cooldown': 2,
'stopper\_min\_load\_time': 3, 'stopper\_post\_detection\_sleep': 1,
'stopper\_threshold\_npts': 50, 'stopper\_threshold\_std': 3.0,
'stopper\_threshold\_v\_step': 1, 'stopper\_timeout': 120}

# AFL.automation.loading.MultiChannelRelay

#### **Classes**

```
MultiChannelRelay()
```

# AFL.automation.loading.MultiChannelRelay.MultiChannelRelay

```
class AFL.automation.loading.MultiChannelRelay.MultiChannelRelay
    Bases: object
    __init__()
```

#### **Methods**

getChannels(channels)

```
__init__()

getChannels(channels)

setChannels(channels)

toggleChannels(channels)

setChannels(channels)
```

```
toggleChannels(channels)

class AFL.automation.loading.MultiChannelRelay.MultiChannelRelay
    setChannels(channels)

    getChannels(channels)

    toggleChannels(channels)
```

## AFL.automation.loading.NE1kSyringePump

#### **Classes**

```
NE1kSyringePump(port, syringe_id_mm, ...[, ...])
SerialDevice(port[, baudrate, timeout, ...])
SyringePump()
```

# AFL.automation.loading.NE1kSyringePump.NE1kSyringePump

```
\begin{tabular}{ll} \textbf{class AFL}. automation.loading. NE1kSyringePump. \textbf{NE1kSyringePump} (port, syringe\_id\_mm, syringe\_volume, baud=9600, \\ & daisy\_chain=None, pumpid=None, \\ & flow\_delay=5) \end{tabular}
```

```
__init__(port, syringe_id_mm, syringe_volume, baud=9600, daisy_chain=None, pumpid=None, flow_delay=5)
```

Initializes and verifies connection to a New Era 1000 syringe pump.

port = serial port reference

Bases: SyringePump

syringe\_id\_mm = syringe inner diameter in mm, used for absolute volume.

(will re-program the pump with this diameter on connection)

syringe\_volume = syringe volume in mL

baud = baudrate for connection

daisy\_chain = used for the 'party-line' mode on these pumps where a string of pumps is on one serial port.

# when setting up daisy chaining:

connect to the first pump on a port with daisy\_chain = False on subsequent pumps, set daisy\_chain to the pump with a hardware connection (the first pump)

or any other pump on the string.

note: when daisy chaining you should probably set pumpid explicitly rather than autodiscovering

as most likely the autodiscovery will return the first pump id each time.

**pumpid = the ID configured in the pump firmware. If not set, will attempt to auto-discover a pump.** setting pumpid will save some time on connection and probably result in more reproducible behavior. Practically mandatory for daisy chain mode.

#### Methods

init(port, syringe_id_mm, syringe_volume)	Initializes and verifies connection to a New Era 1000 syringe pump.
<pre>blockUntilStatusStopped([pollingdelay])</pre>	
dispense(volume[, block, delay])	
<pre>emptySyringe()</pre>	
<pre>getRate()</pre>	
getStatus()	query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
setRate(rate)	
stop()	Abort the current dispense/withdraw action.
withdraw(volume[, block, delay])	

Initializes and verifies connection to a New Era 1000 syringe pump.

port = serial port reference

#### syringe\_id\_mm = syringe inner diameter in mm, used for absolute volume.

(will re-program the pump with this diameter on connection)

syringe volume = syringe volume in mL

baud = baudrate for connection

daisy\_chain = used for the 'party-line' mode on these pumps where a string of pumps is on one serial port.

#### when setting up daisy chaining:

connect to the first pump on a port with daisy\_chain = False on subsequent pumps, set daisy\_chain to the pump with a hardware connection (the first pump)

or any other pump on the string.

# note: when daisy chaining you should probably set pumpid explicitly rather than autodiscovering

as most likely the autodiscovery will return the first pump id each time.

# pumpid = the ID configured in the pump firmware. If not set, will attempt to auto-discover a pump. setting pumpid will save some time on connection and probably result in more reproducible behavior. Practically mandatory for daisy chain mode.

```
stop()
          Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     blockUntilStatusStopped(pollingdelay=0.2)
     getStatus()
          query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
AFL.automation.loading.NE1kSyringePump.SerialDevice
class AFL.automation.loading.NE1kSyringePump.SerialDevice(port, baudrate=19200, timeout=0.5,
                                                                  raw_writes=False)
     Bases: object
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     Methods
       __init__(port[, baudrate, timeout, raw_writes])
      sendCommand(cmd[, response, questionmarkOK,
      ...])
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
AFL.automation.loading.NE1kSyringePump.SyringePump
class AFL.automation.loading.NE1kSyringePump.SyringePump
     Bases: object
     __init__()
```

#### **Methods**

```
_init__()
                     dispense(volume[, block])
                     emptySyringe()
                     getRate(rate)
                     setRate(rate)
                     stop()
                     withdraw(volume[, block])
                 stop()
                 withdraw(volume, block=True)
                 dispense(volume, block=True)
                 setRate(rate)
                 getRate(rate)
                 emptySyringe()
\textbf{class} \ \texttt{AFL}. automation. loading. \texttt{NE1kSyringePump}. \textbf{\textit{NE1kSyringePump}} (port, syringe\_id\_mm, syringe\_id\_mm) (port, syri
                                                                                                                                                                                                                          syringe_volume, baud=9600,
                                                                                                                                                                                                                          daisy_chain=None, pumpid=None,
                                                                                                                                                                                                                          flow_delay=5)
                 __init__(port, syringe_id_mm, syringe_volume, baud=9600, daisy_chain=None, pumpid=None,
                                                 flow_delay=5)
                                Initializes and verifies connection to a New Era 1000 syringe pump.
                                port = serial port reference
                                syringe_id_mm = syringe inner diameter in mm, used for absolute volume.
                                            (will re-program the pump with this diameter on connection)
                                syringe_volume = syringe volume in mL
                                baud = baudrate for connection
                                daisy_chain = used for the 'party-line' mode on these pumps where a string of pumps is on one
                                serial port.
                                            when setting up daisy chaining:
                                                        connect to the first pump on a port with daisy_chain = False on subsequent pumps, set daisy_chain
                                                        to the pump with a hardware connection (the first pump)
                                                              or any other pump on the string.
```

# note: when daisy chaining you should probably set pumpid explicitly rather than autodiscovering

as most likely the autodiscovery will return the first pump id each time.

**pumpid = the ID configured in the pump firmware. If not set, will attempt to auto-discover a pump.** setting pumpid will save some time on connection and probably result in more reproducible behavior. Practically mandatory for daisy chain mode.

#### stop()

Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.

```
withdraw(volume, block=True, delay=True)
dispense(volume, block=True, delay=True)
setRate(rate)
getRate()
emptySyringe()
blockUntilStatusStopped(pollingdelay=0.2)
getStatus()
```

query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)

# AFL.automation.loading.OneSelectorBlowoutSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
OneSelectorBlowoutSampleCell(pump, selector)	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
<pre>TwoSelectorBlowoutSampleCell(pump, selector,)</pre>	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

# AFL.automation.loading.OneSelectorBlowoutSampleCell.Driver

 $\textbf{class} \ \, \textbf{AFL}. a \textbf{utomation.loading.} One \textbf{SelectorBlowoutSampleCell.} \\ \textbf{Driver} (name, \textit{defaults=None}, \\ \textit{overrides=None})$ 

```
Bases: object
__init__(name, defaults=None, overrides=None)
```

# **Methods**

init(name[, defaults, overrides])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	·
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

unqueued()

queued()

quickbar()

```
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

#### AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell

class AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell(pump,

selector,
rinse\_tank\_level=950
waste\_tank\_level=0,
cell\_waste\_tank\_leve
overrides=None)

Bases: TwoSelectorBlowoutSampleCell

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)

```
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0, overrides=None)
```

ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells

thickness = cell path length, to be incorporated into metadata, array with length = ncells

cell state if not 'clean', array with length = ncells

```
\label{pump:apump} \mbox{pump: a pump object supporting withdraw() and dispense() methods}
```

e.g. pump = NE1KSyringePump(port,syringe\_id\_mm,syringe\_volume)

selector: a selector object supporting string-based selectPort() method with options 'catch','cell','rinse','waste','air'

e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1, 'cell':2, 'rinse':3, 'waste':4, 'air':5})

## **Methods**

init(pump, selector[, rinse_tank_level,])	ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
<pre>blowOutCell([cellname, waittime])</pre>	
blowOutCellLegacy([cellname])	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
cellToWaste([cellname])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
drySyringe([blow, waittime])	transfer from air to waste, to push out any residual liquid.

continues on next page

Table 10 – continued from previous page

```
execute(**kwargs)
gather_defaults()
                                                  Gather all inherited static class-level dictionaries
                                                  called default.
get_config(name[, print_console])
get\_configs([print\_console])
get_object(name[, serialize])
get_sample()
loadSample([cellname, sampleVolume])
post_execute(**kwargs)
                                                  Executed after each call to execute
pre_execute(**kwargs)
                                                  Executed before each call to execute
queued()
quickbar()
reset_sample()
reset_tank_levels([rinse, waste, cell_waste])
retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox
rinseAll([cellname])
rinseCatch()
rinseCell([cellname])
rinseCellFlood([cellname])
rinseCellPull([cellname])
rinseSyringe()
setRinseLevel(vol)
setWasteLevel(vol)
set_config(**kwargs)
set_data(data)
                                                  Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
swish(vol)
```

continues on next page

#### Table 10 – continued from previous page

```
transfer(source, dest, vol_source[, vol_dest])
unqueued()
```

#### **Attributes**

```
app
 defaults
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0,
          overrides=None)
    ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
    by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
    thickness = cell path length, to be incorporated into metadata, array with length = ncells
    cell state if not 'clean', array with length = ncells
    pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    selector: a selector object supporting string-based selectPort() method with options
    'catch', 'cell', 'rinse', 'waste', 'air'
        e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
drySyringe(blow=True, waittime=1)
    transfer from air to waste, to push out any residual liquid.
    if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
property app
catchToSyringe(sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
```

```
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
loadSample(cellname='cell', sampleVolume=0)
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
reset_tank_levels(rinse=950, waste=0, cell_waste=0)
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
rinseAll(cellname='cell')
rinseCatch()
rinseCell(cellname='cell')
rinseCellFlood(cellname='cell')
rinseCellPull(cellname='cell')
rinseSyringe()
```

#### AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell

class AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell(pump,

selector, blowselector, rinse\_tank\_level=950 waste\_tank\_level=0, cell\_waste\_tank\_leve

rides=None)

Bases: Driver, SampleCell

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)

# **Methods**

init(pump, selector, blowselector[,])	ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
<pre>blowOutCell([cellname, waittime])</pre>	, ,
blowOutCellLegacy([cellname])	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
<pre>cellToWaste([cellname])</pre>	
<pre>deposit_obj(obj[, uid]) drySyringe([blow, waittime])</pre>	Store an object in the dropbox transfer from air to waste, to push out any residual liquid.
execute(**kwargs)	
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>loadSample([cellname, sampleVolume])</pre>	
post_execute(**kwargs)	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>reset_tank_levels([rinse, waste, cell_waste])</pre>	
retrieve_obj(uid[, delete])	Retrieve an object from the dropbox

continues on next page

Table 11 - continued from previous page

```
rinseAll([cellname])
rinseCatch()
rinseCell([cellname])
rinseCellFlood([cellname])
rinseCellPull([cellname])
rinseSyringe()
setRinseLevel(vol)
setWasteLevel(vol)
set_config(**kwargs)
set_data(data)
                                                 Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
swish(vol)
transfer(source, dest, vol_source[, vol_dest])
unqueued()
```

#### **Attributes**

```
ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
     by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
     thickness = cell path length, to be incorporated into metadata, array with length = ncells
     cell state if not 'clean', array with length = ncells
     pump: a pump object supporting withdraw() and dispense() methods
         e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
     selector: a selector object supporting string-based selectPort() method with options
     'catch', 'cell', 'rinse', 'waste', 'air'
         e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1, 'cell':2, 'rinse':3, 'waste':4, 'air':5})
reset_tank_levels(rinse=950, waste=0, cell_waste=0)
property app
status()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

6.2. Modules 277

• **uid** (str) – The uuid to store the object under

```
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
setWasteLevel(vol)
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
             • data (dict) – Dictionary of data to store in the driver object
             • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
unqueued()
```

# AFL.automation.loading.OneSelectorBlowoutSampleCell.defaultdict

 ${\bf class} \ {\tt AFL.automation.loading.OneSelectorBlowoutSampleCell.} {\bf defaultdict}$ 

Bases: dict

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

#### **Methods**

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
<i>pop</i> (k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

#### **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
$clear() \rightarrow None$ . Remove all items from D.	
$copy() \rightarrow a$ shallow copy of D.	

default\_factory

```
Factory for default value called by __missing__().
      fromkeys(value=None,/)
            Create a new dictionary with keys from iterable and values set to value.
      get(key, default=None, /)
            Return the value for key if key is in the dictionary, else default.
      items() \rightarrow a set-like object providing a view on D's items
      keys() \rightarrow a set-like object providing a view on D's keys
      pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
            If the key is not found, return the default if given; otherwise, raise a KeyError.
      popitem()
            Remove and return a (key, value) pair as a 2-tuple.
            Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.
      setdefault(key, default=None, /)
            Insert key with a value of default if key is not in the dictionary.
            Return the value for key if key is in the dictionary, else default.
      update([E, *F] \rightarrow None. Update D from dict/iterable E and F.
            If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
            method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
      values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell(pump,
                                                                                                                   se-
                                                                                                                   lec-
                                                                                                                   tor,
                                                                                                                   rinse tank level=950
                                                                                                                   waste\_tank\_level=0,
                                                                                                                   cell waste tank leve
                                                                                                                   over-
                                                                                                                   rides=None)
      Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample
      (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell
      where the cell is in the pump line).
      @TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)
      __init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0,
                  overrides=None)
            ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
            by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
            thickness = cell path length, to be incorporated into metadata, array with length = ncells
            cell state if not 'clean', array with length = ncells
            pump: a pump object supporting withdraw() and dispense() methods
                e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
```

# AFL.automation.loading.PneumaticPressureSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
<pre>PneumaticPressureSampleCell(pctrl, relayboard)</pre>	Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
SampleCell()	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

# AFL.automation.loading.PneumaticPressureSampleCell.Driver

```
{\bf class} \  \, {\tt AFL.automation.loading.PneumaticPressureSampleCell.Driver} (name, \textit{defaults=None}, \\ \textit{overrides=None})
```

```
Bases: object
__init__(name, defaults=None, overrides=None)
```

#### **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell

class AFL.automation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell(pctrl,

relayboard, digitalin=None, rinse1\_tank\_level=950, rinse2\_tank\_level=950, waste\_tank\_level=0, load\_stopper=None, robot\_interlock\_host=Noverrides=None)

Bases: Driver, SampleCell

Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.

Driven by a variable-pressure regulator.

```
__init__(pctrl, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950, waste_tank_level=0, load_stopper=None, robot_interlock_host=None, overrides=None)
```

pctrl: a pressure controller object supporting the set\_P method() and the optional p\_ramp() method. e.g. pctrl = DigitalOutPressure Controller(LabJackDigitalOut(...))

# $relay board \ object \ supporting \ string-based \ set Channels () \ method$

required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample' e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1, 'cell':2, 'rinse':3, 'waste':4, 'air':5})

#### **Methods**

init(pctrl, relayboard[, digitalin,])	pctrl: a pressurecontroller object supporting the set_P method() and the optional p_ramp() method.
<pre>advanceSample([load_dest_label])</pre>	Move a sample from one measurement cell to the next
calibrate_sensor()	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>get_sensor_config(**kwargs)</pre>	
	continues on poyt page

continues on next page

Table 12 – continued from previous page

```
loadSample([cellname, sampleVolume, ...])
                                                  Load a sample into the cell
post_execute(**kwargs)
                                                  Executed after each call to execute
pre_execute(**kwargs)
                                                  Executed before each call to execute
primeRinse([waittime])
queued()
quickbar()
read_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
reset_sample()
reset_tank_levels([rinse1, rinse2, waste])
retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox
rinseAll()
rinseCell([cellname])
sensor_reset([sensor_n])
setRinseLevel(vol)
setWasteLevel(vol)
set_config(**kwargs)
set_data(data)
                                                  Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
set_sensor_config(**kwargs)
status()
stopLoad(**kwargs)
unqueued()
```

#### **Attributes**

```
app
 data
 defaults
defaults = {'arm_move_delay': 0.2, 'blowout_pressure': 20,
'external_load_complete_trigger': False, 'load_mode': 'static', 'load_pressure':
2, 'load_timeout': 60, 'ramp_load_duration': 20, 'ramp_load_stop_pressure': 7,
'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
0.5), ('blow', 5)], 'vent_delay': 0.5}
__init__(pctrl, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950,
          waste_tank_level=0, load_stopper=None, robot_interlock_host=None, overrides=None)
     pctrl: a pressurecontroller object supporting the set_P method() and the optional p_ramp() method.
         e.g. pctrl = DigitalOutPressureController(LabJackDigitalOut(...))
     relayboard: a relay board object supporting string-based setChannels() method
         required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
         e.g. selector = SainSmartRelay(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
reset_tank_levels(rinse1=950, rinse2=950, waste=0)
property app
property data
status()
loadSample(cellname='cell', sampleVolume=None, load dest label=")
     Load a sample into the cell
     Params cellname and sampleVolume are kept for backward compat, but are deprecated and unused.
advanceSample(load dest label=")
     Move a sample from one measurement cell to the next
     Params:
         load_dest_label (str, default ''): a 'destination label' for this load.
             labeled sensors will only stop the load if their name is in this destination label. example:
               sensor 1 labeled 'afterSANS' sensor 2 labeled 'beforeSPEC' sensor 3 labeled '' (default)
               advanceSample(load_dest_label='afterSANS') -> sensor 1 or sensor 3 can stop it advance-
               Sample(load dest label='beforeSPEC afterSANS') -> sensor 1, sensor 2, or sensor 3 can
               stop it advanceSample(load_dest_label=") -> only sensor 3 can stop it
stopLoad(**kwargs)
rinseCell(cellname='cell')
rinseAll()
```

```
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
read_sensor()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
```

```
set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
     read_sensor_poll(**kwargs)
     read_sensor_poll_load(**kwargs)
     set_sensor_config(**kwargs)
     get_sensor_config(**kwargs)
     sensor_reset(sensor_n=None)
AFL.automation.loading.PneumaticPressureSampleCell.SampleCell
class AFL.automation.loading.PneumaticPressureSampleCell.SampleCell
     Bases: object
     __init__()
     Methods
       __init__()
      loadSample()
     loadSample()
AFL.automation.loading.PneumaticPressureSampleCell.defaultdict
class AFL.automation.loading.PneumaticPressureSampleCell.defaultdict
     Bases: dict
     defaultdict(default_factory=None, /, [...]) -> dict with default factory
     The default factory is called without arguments to produce a new value when a key is not present, in __getitem__
     only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same
     as if they were passed to the dict constructor, including keyword arguments.
     __init__(*args, **kwargs)
```

# Methods

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
pop(k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

## **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
$clear() \rightarrow None$ . Remove all items from D.	
$copy() \rightarrow a$ shallow copy of D.	
<pre>default_factory     Factory for default value called bymissing()</pre>	
fromkeys(value=None,/) Create a new dictionary with keys from iterable as	nd values set to value.
<pre>get(key, default=None,/)     Return the value for key if key is in the dictionary</pre>	, else default.
$items() \rightarrow a$ set-like object providing a view on D's it	rems
<b>keys()</b> $\rightarrow$ a set-like object providing a view on D's ke	ys

```
pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If the key is not found, return the default if given; otherwise, raise a KeyError.
     popitem()
           Remove and return a (key, value) pair as a 2-tuple.
           Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.
     setdefault(key, default=None, /)
           Insert key with a value of default if key is not in the dictionary.
           Return the value for key if key is in the dictionary, else default.
     update([E, ]^{**F}) \rightarrow None. Update D from dict/iterable E and F.
           If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
           method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
     values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell(pctrl,
                                                                                                         re-
                                                                                                         lay-
                                                                                                         board,
                                                                                                         digi-
                                                                                                         talin=None,
                                                                                                         rinse1 tank level=950,
                                                                                                         rinse2_tank_level=950,
                                                                                                         waste tank level=0,
                                                                                                         load_stopper=None,
                                                                                                         robot interlock host=N
                                                                                                         over-
                                                                                                         rides=None)
     Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
     Driven by a variable-pressure regulator.
     defaults = {'arm_move_delay': 0.2, 'blowout_pressure': 20,
      'external_load_complete_trigger': False, 'load_mode': 'static', 'load_pressure':
     2, 'load_timeout': 60, 'ramp_load_duration': 20, 'ramp_load_stop_pressure': 7,
      'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
     0.5), ('blow', 5)], 'vent_delay': 0.5}
      __init__(pctrl, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950,
                 waste_tank_level=0, load_stopper=None, robot_interlock_host=None, overrides=None)
           pctrl: a pressurecontroller object supporting the set P method() and the optional p ramp() method.
               e.g. pctrl = DigitalOutPressureController(LabJackDigitalOut(...))
           relayboard: a relay board object supporting string-based setChannels() method
               required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
               e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
     reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     property app
     property data
```

```
status()
loadSample(cellname='cell', sampleVolume=None, load_dest_label=")
     Load a sample into the cell
     Params cellname and sampleVolume are kept for backward compat, but are deprecated and unused.
advanceSample(load_dest_label=")
     Move a sample from one measurement cell to the next
     Params:
         load_dest_label (str, default ''): a 'destination label' for this load.
             labeled sensors will only stop the load if their name is in this destination label. example:
               sensor 1 labeled 'afterSANS' sensor 2 labeled 'beforeSPEC' sensor 3 labeled '' (default)
               advanceSample(load_dest_label='afterSANS') -> sensor 1 or sensor 3 can stop it advance-
               Sample(load_dest_label='beforeSPEC afterSANS') -> sensor 1, sensor 2, or sensor 3 can
               stop it advanceSample(load_dest_label=") -> only sensor 3 can stop it
stopLoad(**kwargs)
rinseCell(cellname='cell')
rinseAll()
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
read_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset(sensor_n=None)
```

## AFL.automation.loading.PneumaticSampleCell

## **Classes**

Driver(name[, defaults, overrides])	
<pre>PneumaticSampleCell(pump, relayboard[,])</pre>	Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
SampleCell()	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

# AFL.automation.loading.PneumaticSampleCell.Driver

```
class AFL.automation.loading.PneumaticSampleCell.Driver(name, defaults=None, overrides=None)
    Bases: object
    __init__(name, defaults=None, overrides=None)
```

## Methods

init(name[, defaults, overrides])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	·
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

```
unqueued()
queued()
```

quickbar()

```
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

## AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCell

Bases: Driver, SampleCell

Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.

Driven by a syringe pump.

```
__init__(pump, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950, waste_tank_level=0, load_stopper=None, overrides=None)
```

# $\label{pump:pump:apump} \textbf{pump: a pump object supporting withdraw} () \ \textbf{and dispense} () \ \textbf{methods}$

e.g. pump = NE1KSyringePump(port,syringe\_id\_mm,syringe\_volume)

## relayboard: a relay board object supporting string-based setChannels() method

required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample' e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})

#### **Methods**

init(pump, relayboard[, digitalin,])	pump: a pump object supporting withdraw() and dispense() methods
<pre>calibrate_sensor()</pre>	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>get_sensor_config(**kwargs)</pre>	
<pre>loadSample([cellname, sampleVolume])</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
<pre>primeRinse([waittime])</pre>	

continues on next page

Table 13 – continued from previous page

```
queued()
quickbar()
read_sensor()
read_sensor_pol1(**kwargs)
read_sensor_poll_load(**kwargs)
reset_pump([dispense])
reset_sample()
reset_tank_levels([rinse1, rinse2, waste])
retrieve_obj(uid[, delete])
                                                Retrieve an object from the dropbox
rinseAll()
rinseCell([cellname])
sensor_reset()
setRinseLevel(vol)
setWasteLevel(vol)
set_config(**kwargs)
set_data(data)
                                                Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
set_sensor_config(**kwargs)
status()
stopLoad(**kwargs)
unqueued()
```

#### **Attributes**

```
app
 defaults
defaults = {'air_speed': 30, 'arm_move_delay': 0.2, 'catch_to_cell_vol': 1.15,
'external_load_complete_trigger': False, 'large_dispense_vol': 5, 'load_speed':
2, 'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
0.5), ('blow', 5)], 'vent_delay': 0.5, 'withdraw_vol': 1.5}
__init__(pump, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950,
          waste_tank_level=0, load_stopper=None, overrides=None)
    pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    relayboard: a relay board object supporting string-based setChannels() method
        required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
        e.g. selector = SainSmartRelay(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
reset_pump(dispense=False)
reset_tank_levels(rinse1=950, rinse2=950, waste=0)
property app
status()
loadSample(cellname='cell', sampleVolume=0)
stopLoad(**kwargs)
rinseCell(cellname='cell')
rinseAll()
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
read_sensor()
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
```

```
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
read_sensor_poll(**kwargs)
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
             • data (dict) – Dictionary of data to store in the driver object
             • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
unqueued()
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset()
```

## AFL.automation.loading.PneumaticSampleCell.SampleCell

 $\textbf{class} \ \texttt{AFL}. automation. loading. Pneumatic Sample Cell. \textbf{Sample Cell}$ 

```
Bases: object
__init__()
```

### **Methods**

```
__init__()
loadSample()
```

loadSample()

# AFL.automation.loading.PneumaticSampleCell.defaultdict

class AFL.automation.loading.PneumaticSampleCell.defaultdict

```
Bases: dict
```

defaultdict(default\_factory=None, /, [...]) -> dict with default factory

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

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

# Methods

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
pop(k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

## **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
$clear() \rightarrow None$ . Remove all items from D.	
$copy() \rightarrow a$ shallow copy of D.	
<pre>default_factory     Factory for default value called bymissing()</pre>	
fromkeys(value=None,/) Create a new dictionary with keys from iterable as	nd values set to value.
<pre>get(key, default=None,/)     Return the value for key if key is in the dictionary</pre>	, else default.
$items() \rightarrow a$ set-like object providing a view on D's it	rems
<b>keys()</b> $\rightarrow$ a set-like object providing a view on D's ke	ys

```
pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If the key is not found, return the default if given; otherwise, raise a KeyError.
     popitem()
           Remove and return a (key, value) pair as a 2-tuple.
           Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.
     setdefault(key, default=None, /)
           Insert key with a value of default if key is not in the dictionary.
           Return the value for key if key is in the dictionary, else default.
     update(E, *F) \rightarrow None. Update D from dict/iterable E and F.
           If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
           method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
     values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCell(pump, relayboard,
                                                                                    digitalin=None,
                                                                                    rinse1_tank_level=950,
                                                                                    rinse2 tank level=950,
                                                                                    waste tank level=0,
                                                                                    load_stopper=None,
                                                                                    overrides=None)
     Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.
     Driven by a syringe pump.
     defaults = {'air_speed': 30, 'arm_move_delay': 0.2, 'catch_to_cell_vol': 1.15,
      'external_load_complete_trigger': False, 'large_dispense_vol': 5, 'load_speed':
     2, 'rinse_program': [('rinse1', 5), (None, 2), ('rinse2', 5), ('blow', 5), (None,
     0.5), ('blow', 5)], 'vent_delay': 0.5, 'withdraw_vol': 1.5}
     __init__(pump, relayboard, digitalin=None, rinse1_tank_level=950, rinse2_tank_level=950,
                waste tank level=0, load stopper=None, overrides=None)
           pump: a pump object supporting withdraw() and dispense() methods
               e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
           relayboard: a relay board object supporting string-based setChannels() method
               required channels are 'arm-up', 'arm-down', 'rinse1', 'rinse2', 'blow', 'enable', 'piston-vent', 'postsample'
               e.g. selector = SainSmartRelay(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
     reset_pump(dispense=False)
     reset_tank_levels(rinse1=950, rinse2=950, waste=0)
     property app
     status()
     loadSample(cellname='cell', sampleVolume=0)
     stopLoad(**kwargs)
     rinseCell(cellname='cell')
```

```
rinseAll()
setRinseLevel(vol)
setWasteLevel(vol)
primeRinse(waittime=10)
calibrate_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset()
```

## AFL.automation.loading.PressureController

### **Classes**

PressureController()	Abstract superclass for pressure controllers that provides
	timed dispensing

## AFL.automation.loading.PressureController.PressureController

## ${\bf class} \ \, {\tt AFL.automation.loading.PressureController.} \\ {\bf PressureController.} \\ {\bf Controller.} \\ {\bf Control$

Bases: object

Abstract superclass for pressure controllers that provides timed dispensing

\_\_init\_\_()

## Methods

init()	
<pre>blockUntilStatusStopped([pollingdelay])</pre>	block execution until the controller finishes a dispense
dispenseRunning()	Returns true if a timed dispense is running, false otherwise.
<pre>ramp_dispense(dispense_start_pressure,)</pre>	Perform a pressure dispense with a linear ramp in pressure between <i>dispense_start_pressure</i> and <i>dispense_stop_pressure</i> , stopping after <i>dispense_time</i> .
stop()	Abort the current timed dispense action.
<pre>timed_dispense(dispense_pressure,</pre>	Perform a pressure dispense at pressure dispense_pressure, stopping after dispense_time.

#### **timed\_dispense**(dispense\_pressure, dispense\_time, block=True)

Perform a pressure dispense at pressure dispense\_pressure, stopping after dispense\_time. This dispense can be interrupted by calling self.stop().

#### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

## dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

#### stop()

Abort the current timed dispense action.

## ${\bf class} \ \, {\tt AFL.automation.loading.PressureController.} \\ {\bf PressureController.} \\ {\bf Controller.} \\ {\bf Control$

Abstract superclass for pressure controllers that provides timed dispensing

```
timed_dispense(dispense_pressure, dispense_time, block=True)
```

Perform a pressure dispense at pressure dispense\_pressure, stopping after dispense\_time. This dispense can be interrupted by calling self.stop().

### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

### dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

#### stop()

Abort the current timed dispense action.

### AFL.automation.loading.PressureControllerAsPump

#### **Classes**

```
PressureControllerAsPump(pressure_controller)

SerialDevice(port[, baudrate, timeout, ...])

SyringePump()
```

## AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump

Bases: SyringePump

**\_\_init\_\_**(pressure\_controller, dispense\_pressure=3.5, implied\_flow\_rate=50)

Initialize a pressure controller as a syringe pump.

pressure\_controller: controller to connect to dispense\_pressure: pressure at which dispense should run implied\_flow\_rate: flow rate which should be used to convert to dispense times, mL/min

#### **Methods**

init(pressure_controller[,]) blockUntilStatusStopped([pollingdelay])	Initialize a pressure controller as a syringe pump.
dispense(volume[, block, delay])	
emptySyringe()	
<pre>getRate()</pre>	
getStatus()	query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
setRate(rate)	· · · · · · · · · · · · · · · · · · ·
stop()	Abort the current dispense/withdraw action.
withdraw(volume[, block, delay])	

```
__init__(pressure_controller, dispense_pressure=3.5, implied_flow_rate=50)
```

Initialize a pressure controller as a syringe pump.

pressure\_controller: controller to connect to dispense\_pressure: pressure at which dispense should run implied\_flow\_rate: flow rate which should be used to convert to dispense times, mL/min

## stop()

Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.

```
withdraw(volume, block=True, delay=True)
dispense(volume, block=True, delay=True)
setRate(rate)
getRate()
emptySyringe()
```

```
blockUntilStatusStopped(pollingdelay=0.2)
getStatus()
    query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
```

## AFL.automation.loading.PressureControllerAsPump.SerialDevice

```
class AFL.automation.loading.PressureControllerAsPump.SerialDevice(port, baudrate=19200, timeout=0.5, raw\_writes=False)
```

```
Bases: object __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

### **Methods**

```
__init__(port[, baudrate, timeout, raw_writes])

sendCommand(cmd[, response, questionmarkOK,
...])
```

```
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
```

## AFL.automation.loading.PressureControllerAsPump.SyringePump

```
class AFL.automation.loading.PressureControllerAsPump.SyringePump
    Bases: object
    __init__()
```

## **Methods**

```
__init__()
dispense(volume[, block])

emptySyringe()

getRate(rate)

setRate(rate)

stop()

withdraw(volume[, block])
```

```
stop()
     withdraw(volume, block=True)
     dispense(volume, block=True)
     setRate(rate)
     getRate(rate)
     emptySyringe()
class AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump(pressure_controller,
                                                                                               dis-
                                                                                              pense_pressure=3.5,
                                                                                              plied_flow_rate=50)
     __init__(pressure_controller, dispense_pressure=3.5, implied_flow_rate=50)
          Initialize a pressure controller as a syringe pump.
          pressure_controller: controller to connect to dispense_pressure: pressure at which dispense should run
          implied_flow_rate: flow rate which should be used to convert to dispense times, mL/min
     stop()
          Abort the current dispense/withdraw action. Equivalent to pressing stop button on panel.
     withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     blockUntilStatusStopped(pollingdelay=0.2)
     getStatus()
          query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
```

# AFL.automation.loading.PushPullSelectorSampleCell

## Classes

Driver(name[, defaults, overrides])	
PushPullSelectorSampleCell(pump, selector[,])	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
SampleCell()	
Tubing(specid, length)	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

# ${\bf AFL.} automation. Io ading. Push Pull Selector Sample Cell. Driver$

Bases: object

\_\_init\_\_(name, defaults=None, overrides=None)

### **Methods**

```
_init__(name[, defaults, overrides])
                                                    Store an object in the dropbox
 deposit_obj(obj[, uid])
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

• **uid** (*str*) – The uuid to store the object under

## AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell

class AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell(pump,

selector, ncells=1, thickness=None, catch to sel vol=None, cell\_to\_sel\_vol=None, syringe\_to\_sel\_vol=None, selector internal vol=None, calibrated\_catch\_to\_syringe\_n brated\_syringe\_to\_cell\_vo  $rinse\_speed=50.0$ ,  $load\_speed=10.0,$ rinse flow delay=3.0, load\_flow\_delay=10.0)

Bases: Driver, SampleCell

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector) @TODO: figure out when/where to pull in air to the syringe to make up for extra\_vol\_to\_empty\_ml

```
__init__(pump, selector, ncells=1, thickness=None, catch_to_sel_vol=None, cell_to_sel_vol=None, syringe_to_sel_vol=None, selector_internal_vol=None, calibrated_catch_to_syringe_vol=None, calibrated_syringe_to_cell_vol=None, rinse_speed=50.0, load_speed=10.0, rinse flow delay=3.0, load flow delay=10.0)
```

ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells

thickness = cell path length, to be incorporated into metadata, array with length = ncells

cell state if not 'clean', array with length = ncells

```
pump: a pump object supporting withdraw() and dispense() methods
e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
```

selector: a selector object supporting string-based selectPort() method with options 'catch','cell','rinse','waste','air'

e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})

# Methods

init(pump, selector[, ncells,])	ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
blowOutCell([cellname])	
<pre>blowOutCellForcedAir([cellname, waittime])</pre>	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
cellToWaste([cellname])	
<pre>deposit_obj(obj[, uid]) execute(**kwargs)</pre>	Store an object in the dropbox
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>getParameters()</pre>	
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>loadSample([cellname, sampleVolume])</pre>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
pre_execute(**kwargs)	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
rinseAll([cellname])	
rinseCatch()	
rinseCell([cellname])	
rinseCellFlood([cellname])	
rinseCellPull([cellname])	
	continues on next page

continues on next page

## Table 14 - continued from previous page

#### **Attributes**

```
app
__init__(pump, selector, ncells=1, thickness=None, catch_to_sel_vol=None, cell_to_sel_vol=None,
           syringe_to_sel_vol=None, selector_internal_vol=None, calibrated_catch_to_syringe_vol=None,
           calibrated_syringe_to_cell_vol=None, rinse_speed=50.0, load_speed=10.0,
           rinse_flow_delay=3.0, load_flow_delay=10.0)
     ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
     by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
     thickness = cell path length, to be incorporated into metadata, array with length = ncells
     cell state if not 'clean', array with length = ncells
     pump: a pump object supporting withdraw() and dispense() methods
         e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
     selector: a selector object supporting string-based selectPort() method with options
     'catch', 'cell', 'rinse', 'waste', 'air'
         e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
property app
status()
setParameter(parameter, value)
getParameters()
transfer(source, dest, vol_source, vol_dest=None)
```

```
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
```

```
set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     swish(vol)
     unqueued()
     blowOutCell(cellname='cell')
     blowOutCellForcedAir(cellname='cell', waittime=20)
     rinseCatch()
     rinseAll(cellname='cell')
AFL.automation.loading.PushPullSelectorSampleCell.SampleCell
class AFL.automation.loading.PushPullSelectorSampleCell.SampleCell
     Bases: object
     __init__()
     Methods
       __init__()
      loadSample()
     loadSample()
AFL.automation.loading.PushPullSelectorSampleCell.Tubing
class AFL.automation.loading.PushPullSelectorSampleCell.Tubing(specid, length)
     Bases: object
     __init__(specid, length)
          length is in cm?
```

### **Methods**

init(specid, length)	length is in cm?
volume()	returns volume in mL

### **Attributes**

```
tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in':
```

0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID\_mm': 1, 'OD\_in':

```
__init__(specid, length)
length is in cm?
```

volume()

returns volume in mL

## AFL.automation.loading.PushPullSelectorSampleCell.defaultdict

0.075, 'material': 'Tefzel', 'typeid': 1517}]

class AFL.automation.loading.PushPullSelectorSampleCell.defaultdict

Bases: dict

defaultdict(default\_factory=None, /, [...]) -> dict with default factory

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

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

# Methods

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
pop(k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

## **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
$clear() \rightarrow None$ . Remove all items from D.	
$copy() \rightarrow a$ shallow copy of D.	
<pre>default_factory     Factory for default value called bymissing()</pre>	
fromkeys(value=None,/) Create a new dictionary with keys from iterable as	nd values set to value.
<pre>get(key, default=None,/)     Return the value for key if key is in the dictionary</pre>	, else default.
$items() \rightarrow a$ set-like object providing a view on D's it	rems
<b>keys()</b> $\rightarrow$ a set-like object providing a view on D's ke	ys

popitem()

```
Return the value for key if key is in the dictionary, else default.
      update([E, ]^{**F}) \rightarrow None. Update D from dict/iterable E and F.
           If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
           method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
      values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell(pump,
                                                                                                            selec-
                                                                                                            tor,
                                                                                                            ncells=1,
                                                                                                            thick-
                                                                                                            ness=None.
                                                                                                            catch to sel vol=None,
                                                                                                            cell_to_sel_vol=None,
                                                                                                            ringe_to_sel_vol=None,
                                                                                                            selec-
                                                                                                            tor_internal_vol=None,
                                                                                                            cali-
                                                                                                            brated_catch_to_syringe_v
                                                                                                            cali-
                                                                                                            brated_syringe_to_cell_vo
                                                                                                            rinse\_speed=50.0,
                                                                                                            load\_speed=10.0,
                                                                                                            rinse flow delay=3.0,
                                                                                                            load_flow_delay=10.0)
      Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample
      (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell
      where the cell is in the pump line).
      @TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector) @TODO:
      figure out when/where to pull in air to the syringe to make up for extra_vol_to_empty_ml
      __init__(pump, selector, ncells=1, thickness=None, catch to sel vol=None, cell to sel vol=None,
                 syringe_to_sel_vol=None, selector_internal_vol=None, calibrated_catch_to_syringe_vol=None,
                 calibrated syringe to cell vol=None, rinse speed=50.0, load speed=10.0,
                 rinse flow delay=3.0, load flow delay=10.0)
           ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
           by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
           thickness = cell path length, to be incorporated into metadata, array with length = ncells
           cell state if not 'clean', array with length = ncells
```

 $pop(k[,d]) \rightarrow v$ , remove specified key and return the corresponding value.

Insert key with a value of default if key is not in the dictionary.

Remove and return a (key, value) pair as a 2-tuple.

**setdefault**(*key*, *default=None*, /)

If the key is not found, return the default if given; otherwise, raise a KeyError.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

```
pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    selector: a selector object supporting string-based selectPort() method with options
     'catch','cell','rinse','waste','air'
         e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
property app
status()
setParameter(parameter, value)
getParameters()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCell(cellname='cell')
blowOutCellForcedAir(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
```

## AFL.automation.loading.RSoXSSolutionSampleCell

#### **Classes**

Driver(name[, defaults, overrides])	
RSoXSSolutionSampleCell(pump, selector[,])	Class for a sample cell consisting of a pump and a one-to- many flow selector for a flowrate-limited measurement cell (e.g., a liquid TEM sample holder for RSoXS).
SampleCell()	
Tubing(specid, length)	
defaultdict	defaultdict(default_factory=None, /, [])> dict with default factory

# AFL.automation.loading.RSoXSSolutionSampleCell.Driver

 ${\bf class} \ \, {\tt AFL.automation.loading.RSoXSSolutionSampleCell.Driver} (name, \textit{defaults=None}, \\ \textit{overrides=None})$ 

```
Bases: object
__init__(name, defaults=None, overrides=None)
```

## **Methods**

init(name[, defaults, overrides])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	·
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
post_execute(**kwargs)	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
<pre>set_config(**kwargs)</pre>	
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	

unqueued()

queued()

quickbar()

```
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

## AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell

```
 \textbf{class} \  \, \textbf{AFL}. \textbf{automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell} (pump, selector, \\ rinse\_tank\_level=950, \\ waste\_tank\_level=0, \\ cell\_waste\_tank\_level=0, \\ over-\\ rides=None)
```

Bases: Driver, SampleCell

Class for a sample cell consisting of a pump and a one-to-many flow selector for a flowrate-limited measurement cell (e.g., a liquid TEM sample holder for RSoXS).

The pump draws from any of the sample ports and can quickly purge/rinse itself and the tubing between the loader and the cell. When the pump is connected to the cell, its max rate is limited to very slow (5 ul/min for RSoXS).

```
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0, overrides=None)

ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells

thickness = cell path length, to be incorporated into metadata, array with length = ncells

cell state if not 'clean', array with length = ncells

pump: a pump object supporting withdraw() and dispense() methods

e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)

selector: a selector object supporting string-based selectPort() method with options

'catch','cell','rinse','waste','air'

e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
```

### **Methods**

320

init(pump, selector[, rinse_tank_level,])	ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
blowOutCell([cellname, waittime])	
blowOutCellLegacy([cellname])	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
cellToWaste([cellname])	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
<pre>drySyringe([blow, waittime])</pre>	transfer from air to waste, to push out any residual liquid.
execute(**kwargs)	continues on poyt page

continues on next page

Table 15 - continued from previous page

```
gather_defaults()
                                                  Gather all inherited static class-level dictionaries
                                                  called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
loadSample([cellname, sampleVolume])
post_execute(**kwargs)
                                                  Executed after each call to execute
                                                  Executed before each call to execute
pre_execute(**kwargs)
queued()
quickbar()
reset_sample()
reset_tank_levels([rinse, waste, cell_waste])
retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox
rinseAll([cellname])
rinseCatch()
rinseCell([cellname])
rinseCellFlood([cellname])
rinseCellPull([cellname])
rinseSyringe()
setRinseLevel(vol)
setWasteLevel(vol)
set_config(**kwargs)
set_data(data)
                                                  Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
swish(vol)
transfer(source, dest, vol_source[, vol_dest])
```

continues on next page

### Table 15 – continued from previous page

unqueued()

#### **Attributes**

```
app
 defaults
defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
__init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0,
          overrides=None)
    ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
    by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
    thickness = cell path length, to be incorporated into metadata, array with length = ncells
    cell state if not 'clean', array with length = ncells
    pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    selector: a selector object supporting string-based selectPort() method with options
    'catch', 'cell', 'rinse', 'waste', 'air'
        e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
reset_tank_levels(rinse=950, waste=0, cell waste=0)
property app
status()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
```

```
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
```

```
retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                 uid (str) – The uuid of the file to retrieve
     setWasteLevel(vol)
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
AFL.automation.loading.RSoXSSolutionSampleCell.SampleCell
class AFL.automation.loading.RSoXSSolutionSampleCell.SampleCell
     Bases: object
     __init__()
     Methods
       __init__()
      loadSample()
     loadSample()
AFL.automation.loading.RSoXSSolutionSampleCell.Tubing
class AFL.automation.loading.RSoXSSolutionSampleCell.Tubing(specid, length)
     Bases: object
     __init__(specid, length)
          length is in cm?
```

init(specid, length)	length is in cm?
volume()	returns volume in mL

### **Attributes**

```
tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in':
0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID_mm': 1, 'OD_in':
0.075, 'material': 'Tefzel', 'typeid': 1517}]
__init__(specid, length)
    length is in cm?
volume()
```

# AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict

class AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict

Bases: dict

defaultdict(default\_factory=None, /, [...]) -> dict with default factory

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

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

returns volume in mL

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
pop(k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

# **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
<b>clear()</b> $\rightarrow$ None. Remove all items from D.	
$copy() \rightarrow a$ shallow copy of D.	
default_factory	
Factory for default value called bymissing()  fromkeys(value=None,/)	).
Create a new dictionary with keys from iterable a	and values set to value.
<pre>get(key, default=None, /)</pre>	
Return the value for key if key is in the dictionary <b>items()</b> $\rightarrow$ a set-like object providing a view on D's i	
<b>keys()</b> $\rightarrow$ a set-like object providing a view on D's ke	

```
pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If the key is not found, return the default if given; otherwise, raise a KeyError.
     popitem()
           Remove and return a (key, value) pair as a 2-tuple.
           Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.
     setdefault(key, default=None, /)
           Insert key with a value of default if key is not in the dictionary.
           Return the value for key if key is in the dictionary, else default.
     update(E, *F) \rightarrow None. Update D from dict/iterable E and F.
           If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
           method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
     values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell(pump, selector,
                                                                                             rinse tank level=950,
                                                                                             waste tank level=0,
                                                                                             cell_waste_tank_level=0,
                                                                                             over-
                                                                                             rides=None)
     Class for a sample cell consisting of a pump and a one-to-many flow selector for a flowrate-limited measurement
     cell (e.g., a liquid TEM sample holder for RSoXS).
     The pump draws from any of the sample ports and can quickly purge/rinse itself and the tubing between the
     loader and the cell. When the pump is connected to the cell, its max rate is limited to very slow (5 ul/min for
     RSoXS).
     defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
      'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
      'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
     1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
      'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
      'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
      'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
      'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
     1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
     __init__(pump, selector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0,
                overrides=None)
           ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
           by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
           thickness = cell path length, to be incorporated into metadata, array with length = ncells
           cell state if not 'clean', array with length = ncells
           pump: a pump object supporting withdraw() and dispense() methods
               e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
           selector: a selector object supporting string-based selectPort() method with options
           'catch', 'cell', 'rinse', 'waste', 'air'
               e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
     reset_tank_levels(rinse=950, waste=0, cell_waste=0)
```

```
property app
status()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
setWasteLevel(vol)
```

### AFL.automation.loading.SainSmartRelay

# **Module Attributes**

usbrelay

# AFL.automation.loading.SainSmartRelay.usbrelay

```
AFL.automation.loading.SainSmartRelay.usbrelay = [[b':FE0100200000FF\r\n',
b':FE0100000010F1\r\n'], [b':FE050000000FD\r\n', b':FE050000FF00FE\r\n'],
[b':FE0500010000FC\r\n', b':FE050001FF00FD\r\n'], [b':FE0500020000FB\r\n',
b':FE050002FF00FC\r\n'], [b':FE0500030000FA\r\n', b':FE050003FF00FB\r\n'],
[b':FE0500040000F9\r\n', b':FE050004FF00FA\r\n'], [b':FE0500050000F8\r\n',
b':FE050005FF00F9\r\n'], [b':FE0500060000F7\r\n', b':FE050006FF00F8\r\n'],
[b':FE0500070000F6\r\n', b':FE050007FF00F7\r\n'], [b':FE0500080000F5\r\n',
b':FE050008FF00F6\r\n'], [b':FE0500090000F4\r\n', b':FE050009FF00F5\r\n'],
[b':FE05000A0000F3\r\n', b':FE05000AFF00F4\r\n'], [b':FE05000B0000F2\r\n',
b':FE05000BFF00F3\r\n'], [b':FE05000C0000F1\r\n', b':FE05000CFF00F2\r\n'],
[b':FE05000D0000F0\r\n', b':FE05000DFF00F1\r\n'], [b':FE05000E0000FF\r\n',
b':FE05000EFF00F0\r\n'], [b':FE05000F0000FE\r\n', b':FE05000FF00FF\r\n'],
[b':FE0F00000010020000E1\r\n', b':FE0F0000001002FFFFE3\r\n']]
     "" Sainsmart 16-Channel 9-36v USB Relay Module (CH341 chip)(sku# 101-70-208) 1. Requires CH341 Win-
     dows driver installed (see http://wiki.sainsmart.com/index.php/101-70-208) 2. The hex table provided by Sains-
     mart requires converting to ASCII chars (see 'usbrelay' below) 3. Format of 'usbrelay' two-dimensional array
     is:
          usbrelay = [row][ch-off, ch-on] so that the 2nd index selects ON/OFF value
              while the 1st index selects the array row.
          Example...
              status = usbrelay[0][1] # row-0 (status) stat_ret = usbrelay[0][0] # row-0 (status return) ch_1_on
              = usbrelay[1][1] # row-1 (chan-1 off) ch_1 off = usbrelay[1][0] # row-1 (chan-1 off) ch_16 on =
              usbrelay[16][1] # row-16 (chan-16 on) ch_16_off = usbrelay[16][0] # row-16 (chan-16 off) all_on
              = usbrelay[17][1] # row-17 (all on) all_off = usbrelay[17][0] # row-17 (all off)
```

# **Classes**

667777

```
MultiChannelRelay()

SainSmartRelay(relaylabels, serial_port[, ...])

SerialDevice(port[, baudrate, timeout, ...])
```

#### AFL.automation.loading.SainSmartRelay.MultiChannelRelay

```
class AFL.automation.loading.SainSmartRelay.MultiChannelRelay
    Bases: object
    __init__()
```

```
__init__()

getChannels(channels)

setChannels(channels)

toggleChannels(channels)

setChannels(channels)

toggleChannels(channels)
```

# AFL.automation.loading.SainSmartRelay.SainSmartRelay

```
class AFL.automation.loading.SainSmartRelay.SainSmartRelay(relaylabels, serial_port, timeout=0.5)
    Bases: MultiChannelRelay, SerialDevice
    __init__(relaylabels, serial_port, timeout=0.5)
    Init connection to a SainSmart 16-channel USB relay module.
    Params: relaylabels (dict):
        mapping of port id to load name, e.g. {0:'arm_up',1:'arm_down'}
    serial_port (str):
        port to connect to
```

# **Methods**

330

```
__init__(relaylabels, serial_port[, timeout])

getChannels([asid])

sendCommand(cmd[, response, questionmarkOK,
...])

setAllChannels(channels)

Write a value (True, False) to the channels specified in channels

toggleChannels(channels)
```

```
__init__(relaylabels, serial_port, timeout=0.5)
```

Init connection to a SainSmart 16-channel USB relay module.

Params: relaylabels (dict):

```
mapping of port id to load name, e.g. {0:'arm_up',1:'arm_down'}
           serial port (str):
               port to connect to
     setAllChannelsOff()
     setChannels(channels)
           Write a value (True, False) to the channels specified in channels
           Parameters: channels (dict):
               dict of channels, keys as either str (name) or int (id) to write to, vals as the value to write
     getChannels(asid=False)
           Read the current state of all channels
           Parameters: asid (bool,default false): Dict keys should simply be the id, not the name.
           Returns: (dict) key:value mappings of state.
     toggleChannels(channels)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
AFL.automation.loading.SainSmartRelay.SerialDevice
class AFL.automation.loading.SainSmartRelay.SerialDevice(port, baudrate=19200, timeout=0.5,
                                                                     raw_writes=False)
     Bases: object
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     Methods
       __init__(port[, baudrate, timeout, raw_writes])
       sendCommand(cmd[, response, questionmarkOK,
       ...])
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
class AFL.automation.loading.SainSmartRelay.SainSmartRelay(relaylabels, serial_port, timeout=0.5)
     __init__(relaylabels, serial_port, timeout=0.5)
           Init connection to a SainSmart 16-channel USB relay module.
           Params: relaylabels (dict):
               mapping of port id to load name, e.g. {0:'arm_up',1:'arm_down'}
           serial_port (str):
               port to connect to
```

```
setAllChannelsOff()
     setChannels(channels)
          Write a value (True, False) to the channels specified in channels
          Parameters: channels (dict):
              dict of channels, keys as either str (name) or int (id) to write to, vals as the value to write
     getChannels(asid=False)
          Read the current state of all channels
          Parameters: asid (bool,default false): Dict keys should simply be the id, not the name.
          Returns: (dict) key:value mappings of state.
     toggleChannels(channels)
AFL.automation.loading.SainSmartRelay.usbrelay = [[b':FE0100200000FF\r\n',
b':FE0100000010F1\r\n'], [b':FE050000000FD\r\n', b':FE050000FF00FE\r\n'],
[b':FE0500010000FC\r\n', b':FE050001FF00FD\r\n'], [b':FE0500020000FB\r\n',
b':FE050002FF00FC\r\n'], [b':FE0500030000FA\r\n', b':FE050003FF00FB\r\n'],
[b':FE0500040000F9\r\n', b':FE050004FF00FA\r\n'], [b':FE0500050000F8\r\n',
b':FE050005FF00F9\r\n'], [b':FE0500060000F7\r\n', b':FE050006FF00F8\r\n'],
[b':FE0500070000F6\r\n', b':FE050007FF00F7\r\n'], [b':FE0500080000F5\r\n',
b':FE050008FF00F6\r\n'], [b':FE0500090000F4\r\n', b':FE050009FF00F5\r\n'],
[b':FE05000A0000F3\r\n', b':FE05000AFF00F4\r\n'], [b':FE05000B0000F2\r\n',
b':FE05000BFF00F3\r\n'], [b':FE05000C0000F1\r\n', b':FE05000CFF00F2\r\n'],
[b':FE05000D0000F0\r\n', b':FE05000DFF00F1\r\n'], [b':FE05000E0000FF\r\n',
b':FE05000EFF00F0\r\n'], [b':FE05000F0000FE\r\n', b':FE05000FF00FF\r\n'],
[b':FE0F00000010020000E1\r\n', b':FE0F0000001002FFFFE3\r\n']]
     "" Sainsmart 16-Channel 9-36v USB Relay Module (CH341 chip)(sku# 101-70-208) 1. Requires CH341 Win-
     dows driver installed (see http://wiki.sainsmart.com/index.php/101-70-208) 2. The hex table provided by Sains-
     mart requires converting to ASCII chars (see 'usbrelay' below) 3. Format of 'usbrelay' two-dimensional array
          usbrelay = [row][ch-off, ch-on] so that the 2nd index selects ON/OFF value
              while the 1st index selects the array row.
          Example...
              status = usbrelay[0][1] # row-0 (status) stat_ret = usbrelay[0][0] # row-0 (status return) ch_1_on
              = usbrelay[1][1] # row-1 (chan-1 off) ch 1 off = <math>usbrelay[1][0] # row-1 (chan-1 off) ch 16 on =
              usbrelay[16][1] # row-16 (chan-16 on) ch_16_off = usbrelay[16][0] # row-16 (chan-16 off) all_on
              = usbrelay[17][1] # row-17 (all on) all_off = usbrelay[17][0] # row-17 (all off)
     662222
AFL.automation.loading.SampleCell
```

#### Classes

```
SampleCell()
```

# AFL.automation.loading.SampleCell.SampleCell

```
class AFL.automation.loading.SampleCell.SampleCell
    Bases: object
    __init__()
```

#### **Methods**

```
__init__()
loadSample()
```

### loadSample()

class AFL.automation.loading.SampleCell.SampleCell

loadSample()

# AFL.automation.loading.Sensor

#### **Classes**

```
DummySensor1([period, minval, maxval])

DummySensor2([period, hi_value, lo_time, ...])

Sensor([address, channel])
```

# AFL.automation.loading.Sensor.DummySensor1

```
class AFL.automation.loading.Sensor.DummySensor1(period=2, minval=-5, maxval=5)
    Bases: Thread
    __init__(period=2, minval=-5, maxval=5)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init([period, minval, maxval])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

# **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

### **\_\_init\_\_**(period=2, minval=-5, maxval=5)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

# read()

#### terminate()

# run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

# getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

# property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

# is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

# property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

# start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

# AFL.automation.loading.Sensor.DummySensor2

class AFL.automation.loading.Sensor.DummySensor2(period=0.1, hi\_value=5, lo\_time=15, hi\_time=2)

Bases: Thread

```
__init__(period=0.1, hi_value=5, lo_time=15, hi_time=2)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init([period, hi_value, lo_time, hi_time])	This constructor should always be called with keyword arguments.
<pre>driver_status()</pre>	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

```
__init__(period=0.1, hi_value=5, lo_time=15, hi_time=2)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread. $\_$ init $\_$ ()) before doing anything else to the thread.

driver\_status()

read()

terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

# property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

# is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

```
setDaemon(daemonic)
```

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

# AFL.automation.loading.Sensor.Sensor

```
class AFL.automation.loading.Sensor.Sensor(address=1, channel=0)
    Bases: object
    __init__(address=1, channel=0)
```

# **Methods**

```
__init__([address, channel])
calibrate()
read()

__init__(address=1, channel=0)
calibrate()
read()

class AFL.automation.loading.Sensor.Sensor(address=1, channel=0)
    __init__(address=1, channel=0)
calibrate()
read()

class AFL.automation.loading.Sensor.DummySensor1(period=2, minval=-5, maxval=5)
```

```
__init__(period=2, minval=-5, maxval=5)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### read()

#### terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class AFL.automation.loading.Sensor.DummySensor2(period=0.1, hi\_value=5, lo\_time=15, hi\_time=2)

```
__init__(period=0.1, hi value=5, lo time=15, hi time=2)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

### driver\_status()

### read()

### terminate()

# run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

341

# AFL.automation.loading.SensorCallbackThread

### **Classes**

```
LoaderCommunication([load_client, load_object])

SensorCallbackThread(poll[, period, daemon, ...])

SimpleThresholdCB(poll, period[, window, ...])

StopLoadCBv1(poll, period, load_client[, ...])

StopLoadCBv2(poll, period[, load_client, ...])
```

# AFL.automation.loading.SensorCallbackThread.LoaderCommunication

```
Bases: object
__init__(load_client=None, load_object=None)
```

### **Methods**

```
__init__([load_client, load_object])

getServerState()

stopLoad()

__init__(load_client=None, load_object=None)

getServerState()

stopLoad()
```

# AFL.automation.loading.SensorCallbackThread.SensorCallbackThread

```
 \textbf{class} \  \, \textbf{AFL}. \textbf{automation.loading.SensorCallbackThread.SensorCallbackThread} (poll, period = 0.1, \\ daemon = True, \\ filepath = None, \\ data = None, \\ sensorlabel = ")
```

Bases: Thread

\_\_init\_\_(poll, period=0.1, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

init(poll[, period, daemon, filepath,])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal(signal)</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

# **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(poll, period=0.1, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

### update\_status(value)

#### terminate()

### process\_signal(signal)

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

# property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

# is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call

is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

# property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

# property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

# start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

# AFL. automation. loading. Sensor Callback Thread. Simple Threshold CB

# Bases: SensorCallbackThread

```
__init__(poll, period, window=5, threshold=1)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init(poll, period[, window, threshold])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal()</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

# **\_\_init\_\_**(poll, period, window=5, threshold=1)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

# process\_signal()

# property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

# isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

# property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

# start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### terminate()

update\_status(value)

# AFL.automation.loading.SensorCallbackThread.StopLoadCBv1

class AFL.automation.loading.SensorCallbackThread.StopLoadCBv1(poll, period, load\_client,

threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

#### Bases: SensorCallbackThread

\_\_init\_\_(poll, period, load\_client, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init(poll, period, load_client[,])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal()</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(poll, period, load\_client, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

### process\_signal()

### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

# run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

# start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### terminate()

update\_status(value)

# AFL.automation.loading.SensorCallbackThread.StopLoadCBv2

class AFL.automation.loading.SensorCallbackThread.StopLoadCBv2(poll, period, load\_client=None,

load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

#### Bases: SensorCallbackThread

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

init(poll, period[, load_client,])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
<pre>process_signal()</pre>	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	
update_status(value)	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

process\_signal()

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

# getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

# property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

# is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

# property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### terminate()

```
update_status(value)
```

```
class AFL.automation.loading.SensorCallbackThread.SensorCallbackThread(poll, period = 0.1, daemon = True, filepath = None, data = None, sensorlabel = ")
```

\_\_init\_\_(poll, period=0.1, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

```
update_status(value)
terminate()
process_signal(signal)
```

```
run()
```

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

class AFL.automation.loading.SensorCallbackThread.StopLoadCBv1(poll, period, load\_client,

threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel="')

\_\_init\_\_(poll, period, load\_client, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

# process\_signal()

class AFL.automation.loading.SensorCallbackThread.StopLoadCBv2(poll, period, load\_client=None,

load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

\_\_init\_\_(poll, period, load\_client=None, load\_object=None, threshold\_npts=20, threshold\_v\_step=1, threshold\_std=2.5, timeout=120, min\_load\_time=3, loadstop\_cooldown=2, post\_detection\_sleep=0.2, baseline\_duration=2, trigger\_on\_end=False, instatrigger=True, daemon=True, filepath=None, data=None, sensorlabel=")

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

```
process_signal()
```

```
__init__(poll, period, window=5, threshold=1)
```

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

```
process_signal()
```

```
__init__(load_client=None, load_object=None)
getServerState()
stopLoad()
```

# AFL.automation.loading.SensorPollingThread

#### **Classes**

SensorPollingThread(sensor[, period, ...])

# AFL.automation.loading.SensorPollingThread.SensorPollingThread

 $\textbf{class} \ \texttt{AFL.automation.loading.SensorPollingThread}. \textbf{SensorPollingThread} (\textit{sensor}, \textit{period} = 0.1, \texttt{automation.loading.SensorPollingThread}) (\textit{sensor}, \textit{perio$ 

callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

Bases: Thread

\_\_init\_\_(sensor, period=0.1, callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

init(sensor[, period, callback,])	This constructor should always be called with keyword arguments.
alive()	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
read_load_buffer()	
reset_load_buffer()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

\_\_init\_\_(sensor, period=0.1, callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### read()

#### read\_load\_buffer()

reset\_load\_buffer()

### terminate()

alive()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

```
class AFL.automation.loading.SensorPollingThread.SensorPollingThread(sensor, period=0.1,
                                                                                     callback=None,
                                                                                     hv pipe=None,
                                                                                     window=None,
                                                                                    filename=None,
                                                                                     daemon=True,
                                                                                     data=None)
     __init__(sensor, period=0.1, callback=None, hv_pipe=None, window=None, filename=None,
                daemon=True, data=None)
           This constructor should always be called with keyword arguments. Arguments are:
           group should be None; reserved for future extension when a ThreadGroup class is implemented.
           target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
           name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a
           small decimal number.
           args is a list or tuple of arguments for the target invocation. Defaults to ().
           kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.
           If a subclass overrides the constructor, it must make sure to invoke the base class constructor
           (Thread.__init__()) before doing anything else to the thread.
     read()
     read_load_buffer()
     reset_load_buffer()
     terminate()
     alive()
     run()
           Method representing the thread's activity.
           You may override this method in a subclass. The standard run() method invokes the callable object passed
```

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

### AFL.automation.loading.SerialDevice

#### **Classes**

```
SerialDevice(port[, baudrate, timeout, ...])
```

### AFL.automation.loading.SerialDevice.SerialDevice

 $\begin{tabular}{ll} \textbf{class} & \textbf{AFL}. \textbf{automation.} \textbf{loading.} \textbf{SerialDevice}. \textbf{SerialDevice}(port, baudrate=19200, timeout=0.5, \\ & raw\_writes=False) \end{tabular}$ 

Bases: object

**\_\_init\_\_**(port, baudrate=19200, timeout=0.5, raw\_writes=False)

#### **Methods**

```
__init__(port[, baudrate, timeout, raw_writes])

sendCommand(cmd[, response, questionmarkOK,
...])
```

**\_\_init\_\_**(port, baudrate=19200, timeout=0.5, raw\_writes=False)

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

### **Exceptions**

SerialCommsException	Raised when the system receives a serial response it can't
	parse, likely a garbled line

### AFL.automation.loading.SerialDevice.SerialCommsException

exception AFL.automation.loading.SerialDevice.SerialCommsException

Raised when the system receives a serial response it can't parse, likely a garbled line

```
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

**sendCommand**(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)

### AFL.automation.loading.SyringePump

#### **Classes**

SyringePump()

### AFL.automation.loading.SyringePump.SyringePump

```
class AFL.automation.loading.SyringePump.SyringePump
    Bases: object
    __init__()
```

#### **Methods**

```
__init__()
dispense(volume[, block])
emptySyringe()
getRate(rate)
setRate(rate)
stop()
withdraw(volume[, block])
```

```
stop()
withdraw(volume, block=True)
dispense(volume, block=True)
setRate(rate)
getRate(rate)
emptySyringe()
class AFL.automation.loading.SyringePump.SyringePump
stop()
withdraw(volume, block=True)
dispense(volume, block=True)
setRate(rate)
getRate(rate)
emptySyringe()
```

### AFL.automation.loading.Tubing

#### Classes

```
Tubing(specid, length)
```

### AFL.automation.loading.Tubing.Tubing

```
class AFL.automation.loading.Tubing.Tubing(specid, length)
    Bases: object
    __init__(specid, length)
    length is in cm?
```

#### **Methods**

```
__init__(specid, length) length is in cm?
volume() returns volume in mL
```

#### **Attributes**

```
tubing
    tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
     'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
     'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in':
    0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID_mm': 1, 'OD_in':
    0.075, 'material': 'Tefzel', 'typeid': 1517}]
    __init__(specid, length)
        length is in cm?
    volume()
        returns volume in mL
class AFL.automation.loading.Tubing.Tubing(specid, length)
    tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
     'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
     'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in':
    0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID_mm': 1, 'OD_in':
    0.075, 'material': 'Tefzel', 'typeid': 1517}]
    __init__(specid, length)
        length is in cm?
```

### volume()

returns volume in mL

### AFL.automation.loading.TwoSelectorBlowoutSampleCell

#### **Classes**

Driver(name[, defaults, overrides])		
SampleCell()		
Tubing(specid, length)		
TwoSelectorBlowoutSampleCell(pump,)	selector,	Class for a sample cell consisting of a pump and a one- to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a sepa- rate selector channel (in contrast to an inline selector cell where the cell is in the pump line).
defaultdict		defaultdict(default_factory=None, /, [])> dict with default factory

### ${\bf AFL}. automation. loading. Two Selector Blowout Sample Cell. Driver$

Bases: object

\_\_init\_\_(name, defaults=None, overrides=None)

#### **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

### AFL.automation.loading.TwoSelectorBlowoutSampleCell.SampleCell

```
class AFL.automation.loading.TwoSelectorBlowoutSampleCell.SampleCell
     Bases: object
     __init__()
```

#### **Methods**

```
__init__()
loadSample()
```

loadSample()

### AFL.automation.loading.TwoSelectorBlowoutSampleCell.Tubing

```
class AFL.automation.loading.TwoSelectorBlowoutSampleCell.Tubing(specid, length)
    Bases: object
    __init__(specid, length)
    length is in cm?
```

#### **Methods**

```
__init__(specid, length) length is in cm?
volume() returns volume in mL
```

#### **Attributes**

returns volume in mL

```
tubing = [{'IDEXpart': 1530, 'ID_mm': 1.575, 'OD_in': 0.125, 'material':
'Tefzel', 'typeid': 1530}, {'IDEXpart': 1529, 'ID_mm': 0.254, 'OD_in': 0.075,
'material': 'Tefzel', 'typeid': 1529}, {'IDEXpart': 0, 'ID_mm': 2.92, 'OD_in':
0.1875, 'material': 'PVC', 'typeid': 1}, {'IDEXpart': 1517, 'ID_mm': 1, 'OD_in':
0.075, 'material': 'Tefzel', 'typeid': 1517}]
__init__(specid, length)
    length is in cm?
volume()
```

### AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell

class AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell(pump,

selector, blowselector, rinse\_tank\_level=950 waste\_tank\_level=0, cell\_waste\_tank\_leve overrides=None)

Bases: Driver, SampleCell

Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell where the cell is in the pump line).

@TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)

```
__init__(pump, selector, blowselector, rinse_tank_level=950, waste_tank_level=0, cell_waste_tank_level=0, overrides=None)
```

ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells

thickness = cell path length, to be incorporated into metadata, array with length = ncells

cell state if not 'clean', array with length = ncells

### pump: a pump object supporting withdraw() and dispense() methods

e.g. pump = NE1KSyringePump(port,syringe\_id\_mm,syringe\_volume)

# selector: a selector object supporting string-based selectPort() method with options 'catch','cell','rinse','waste','air'

e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})

#### **Methods**

init(pump, selector, blowselector[,])	ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
blowOutCell([cellname, waittime])	
blowOutCellLegacy([cellname])	
<pre>catchToSyringe([sampleVolume])</pre>	
catchToWaste([sampleVolume])	

continues on next page

Table 16 – continued from previous page

Table 16 – continue	d from previous page
<pre>cellToWaste([cellname])</pre>	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
drySyringe([blow, waittime])	transfer from air to waste, to push out any residual liquid.
execute(**kwargs)	
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>get_config(name[, print_console])</pre>	
<pre>get_configs([print_console])</pre>	
<pre>get_object(name[, serialize])</pre>	
<pre>get_sample()</pre>	
<pre>loadSample([cellname, sampleVolume])</pre>	
<pre>post_execute(**kwargs) pre_execute(**kwargs)</pre>	Executed after each call to execute  Executed before each call to execute
queued()	Executed before each can to execute
quickbar()	
reset_sample()	
<pre>reset_tank_levels([rinse, waste, cell_waste])</pre>	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
rinseAll([cellname])	
rinseCatch()	
rinseCell([cellname])	
rinseCellFlood([cellname])	
rinseCellPull([cellname])	
rinseSyringe()	
setRinseLevel(vol)	
setWasteLevel(vol)	
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object
set_object([serialized])	
	continues on next page

continues on next page

### Table 16 - continued from previous page

```
set_sample(sample_name[, sample_uuid])
status()
swish(vol)
transfer(source, dest, vol_source[, vol_dest])
unqueued()
```

#### **Attributes**

```
app
 defaults
defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
__init__(pump, selector, blowselector, rinse tank level=950, waste tank level=0,
          cell waste tank level=0, overrides=None)
    ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
    by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
    thickness = cell path length, to be incorporated into metadata, array with length = ncells
    cell state if not 'clean', array with length = ncells
    pump: a pump object supporting withdraw() and dispense() methods
        e.g. pump = NE1KSyringePump(port,syringe_id_mm,syringe_volume)
    selector: a selector object supporting string-based selectPort() method with options
    'catch', 'cell', 'rinse', 'waste', 'air'
        e.g. selector = ViciMultiposSelector(port,portlabels={'catch':1,'cell':2,'rinse':3,'waste':4,'air':5})
reset_tank_levels(rinse=950, waste=0, cell_waste=0)
property app
status()
transfer(source, dest, vol source, vol dest=None)
catchToSyringe(sampleVolume=0)
```

```
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
```

Chapter 6. Reference

by subclasses.

```
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
setWasteLevel(vol)
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
             • data (dict) – Dictionary of data to store in the driver object
             • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
unqueued()
```

### AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultdict

### ${\bf class} \ {\tt AFL.automation.loading.TwoSelectorBlowoutSampleCell.} {\bf defaultdict}$

```
Bases: dict
defaultdict(default_factory=None, /, [...]) -> dict with default factory
```

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

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

### Methods

init(*args, **kwargs)	
clear()	
copy()	
<pre>fromkeys([value])</pre>	Create a new dictionary with keys from iterable and values set to value.
<pre>get(key[, default])</pre>	Return the value for key if key is in the dictionary, else default.
<pre>items()</pre>	
keys()	
pop(k[,d])	If the key is not found, return the default if given; otherwise, raise a KeyError.
<pre>popitem()</pre>	Remove and return a (key, value) pair as a 2-tuple.
setdefault(key[, default])	Insert key with a value of default if key is not in the dictionary.
update([E, ]**F)	If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$
values()	

### **Attributes**

default_factory	Factory for default value called bymissing().
init(*args, **kwargs)	
<b>clear()</b> $\rightarrow$ None. Remove all items from D.	
$copy() \rightarrow a \text{ shallow copy of D.}$	
default_factory	
Factory for default value called bymissing(	).
<pre>fromkeys(value=None,/)</pre>	
Create a new dictionary with keys from iterable	and values set to value.
<pre>get(key, default=None, /)</pre>	
Return the value for key if key is in the dictionar	y, else default.
<b>items()</b> $\rightarrow$ a set-like object providing a view on D's	items
<b>keys()</b> $\rightarrow$ a set-like object providing a view on D's k	eys

```
pop(k[,d]) \rightarrow v, remove specified key and return the corresponding value.
           If the key is not found, return the default if given; otherwise, raise a KeyError.
     popitem()
           Remove and return a (key, value) pair as a 2-tuple.
           Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.
     setdefault(key, default=None, /)
           Insert key with a value of default if key is not in the dictionary.
           Return the value for key if key is in the dictionary, else default.
     update(E, **F) \rightarrow None. Update D from dict/iterable E and F.
           If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys()
           method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
     values() \rightarrow an object providing a view on D's values
class AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell(pump,
                                                                                                          se-
                                                                                                          lec-
                                                                                                          tor,
                                                                                                          blows-
                                                                                                          e-
                                                                                                          lec-
                                                                                                          tor.
                                                                                                          rinse tank level=950
                                                                                                          waste\_tank\_level=0,
                                                                                                          cell waste tank leve
                                                                                                          over-
                                                                                                          rides=None)
     Class for a sample cell consisting of a pump and a one-to-many flow selector where the pump line holds sample
     (pulling and pushing as necessary) with a cell on a separate selector channel (in contrast to an inline selector cell
     where the cell is in the pump line).
      @TODO: write support for multiple cells on separate channels (up to 6 cells on a 10-position selector)
     defaults = {'blow_out_vol': 6, 'calibrated_catch_to_syringe_vol': 1.1,
      'calibrated_syringe_to_cell_vol': 3.2, 'catch_empty_ffvol': 2,
      'catch_to_selector_vol': 0.8796459430051422, 'cell_to_selector_vol':
     1.9351768777756622, 'dry_vol_ml': 5, 'load_flow_delay': 10.0, 'load_speed': 10.0,
      'ncells': 1, 'nrinses_catch': 2, 'nrinses_cell': 1, 'nrinses_cell_flood': 2,
      'nrinses_syringe': 2, 'rinse_flow_delay': 3.0, 'rinse_prime_vol': 3,
      'rinse_speed': 50.0, 'rinse_vol_catch_ml': 2, 'rinse_vol_ml': 3,
      'selector_internal_vol': 0.0005067074790974977, 'syringe_to_selector_vol':
     1.1625376729937205, 'thickness': None, 'to_waste_vol': 1}
     __init__(pump, selector, blowselector, rinse tank level=950, waste tank level=0,
                cell waste tank level=0, overrides=None)
           ncells = number of connected cells (up to 6 cells with a 10-position flow selector, with four positions taken
           by load port, rinse, waste, and air) Name = the cell name, array with length = ncells
           thickness = cell path length, to be incorporated into metadata, array with length = ncells
           cell state if not 'clean', array with length = ncells
           pump: a pump object supporting withdraw() and dispense() methods
```

6.2. Modules 373

e.g. pump = NE1KSyringePump(port,syringe\_id\_mm,syringe\_volume)

```
selector: a selector object supporting string-based selectPort() method with options
     'catch','cell','rinse','waste','air'
         e.g. selector = ViciMultiposSelector(port,portlabels={ 'catch':1, 'cell':2, 'rinse':3, 'waste':4, 'air':5})
reset_tank_levels(rinse=950, waste=0, cell waste=0)
property app
status()
transfer(source, dest, vol_source, vol_dest=None)
catchToSyringe(sampleVolume=0)
loadSample(cellname='cell', sampleVolume=0)
catchToWaste(sampleVolume=0.0)
cellToWaste(cellname='cell')
rinseSyringe()
drySyringe(blow=True, waittime=1)
     transfer from air to waste, to push out any residual liquid.
     if blow is True, additionally use a 1 s pulse of nitrogen to clear the syringe transfer line.
rinseCell(cellname='cell')
rinseCellPull(cellname='cell')
rinseCellFlood(cellname='cell')
swish(vol)
blowOutCellLegacy(cellname='cell')
blowOutCell(cellname='cell', waittime=20)
rinseCatch()
rinseAll(cellname='cell')
setRinseLevel(vol)
setWasteLevel(vol)
```

### AFL.automation.loading.UltimusVPressureController

### **Classes**

PressureController()	Abstract superclass for pressure controllers that provides timed dispensing
<pre>UltimusVPressureController(port[, baud])</pre>	

### AFL.automation.loading.UltimusVPressureController.PressureController

### class AFL.automation.loading.UltimusVPressureController.PressureController

Bases: object

Abstract superclass for pressure controllers that provides timed dispensing

\_\_init\_\_()

#### **Methods**

init()	
<pre>blockUntilStatusStopped([pollingdelay])</pre>	block execution until the controller finishes a dispense
dispenseRunning()	Returns true if a timed dispense is running, false otherwise.
<pre>ramp_dispense(dispense_start_pressure,)</pre>	Perform a pressure dispense with a linear ramp in pressure between <i>dispense_start_pressure</i> and <i>dispense_stop_pressure</i> , stopping after <i>dispense_time</i> .
stop()	Abort the current timed dispense action.
<pre>timed_dispense(dispense_pressure,</pre>	Perform a pressure dispense at pressure dispense_pressure, stopping after dispense_time.

### timed\_dispense(dispense\_pressure, dispense\_time, block=True)

Perform a pressure dispense at pressure dispense\_pressure, stopping after dispense\_time. This dispense can be interrupted by calling self.stop().

### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

### dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

## stop()

Abort the current timed dispense action.

### AFL.automation.loading.UltimusVPressureController.UltimusVPressureController

 $\textbf{class} \ \texttt{AFL}. \textbf{automation.} loading. \textbf{UltimusVPressureController.} \textbf{UltimusVPressureController} (port, port, p$ 

baud=115200)

#### **Methods**

init(port[, baud])		Initializes a DigitalOutPressureController
<pre>blockUntilStatusStopped([pollingdelay])</pre>		block execution until the controller finishes a dispense
char_count(cmd)		
compute_checksum(cmd)		
dispenseRunning()		Returns true if a timed dispense is running, false otherwise.
<pre>package_cmd(cmd)</pre>		
<pre>ramp_dispense(dispense_start_pressure,)</pre>		Perform a pressure dispense with a linear ramp in pressure between <i>dispense_start_pressure</i> and <i>dispense_stop_pressure</i> , stopping after <i>dispense_time</i> .
send_command(cmd)		
<pre>set_P(pressure)</pre>		pressure: pressure to set in psi
stop()		Abort the current timed dispense action.
<pre>timed_dispense(dispense_pressure, pense_time)</pre>	dis-	Perform a pressure dispense at pressure dispense_pressure, stopping after dispense_time.

### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

### dispenseRunning()

Returns true if a timed dispense is running, false otherwise.

Perform a pressure dispense with a linear ramp in pressure between *dispense\_start\_pressure* and *dispense\_stop\_pressure*, stopping after *dispense\_time*. This dispense can be interrupted by calling self.stop(). If const\_time is set, the last *const\_time* seconds of the dispense will be at constant pressure, with the ramp occurring in the remaining time.

```
stop()
```

Abort the current timed dispense action.

```
timed_dispense(dispense_pressure, dispense_time, block=True)
```

Perform a pressure dispense at pressure dispense\_pressure, stopping after dispense\_time. This dispense can be interrupted by calling self.stop().

 $\textbf{class} \ \texttt{AFL}. \textbf{automation.} loading. \textbf{Ultimus VP} ressure \texttt{Controller.} \textbf{Ultimus VP} ressure \texttt{Controller} (\textit{port}, \textit{port}, \textit$ 

baud=115200)

```
__init__(port, baud=115200)
Initializes a DigitalOutPressureController
Params:
        port (str): serial port to use

compute_checksum(cmd)

char_count(cmd)

package_cmd(cmd)

send_command(cmd)

set_P(pressure)
        pressure: pressure to set in psi
```

### AFL.automation.loading.ViciMultiposSelector

#### **Classes**

```
FlowSelector()

SerialDevice(port[, baudrate, timeout, ...])

ViciMultiposSelector(port[, baudrate, ...])
```

### AFL.automation.loading.ViciMultiposSelector.FlowSelector

```
class AFL.automation.loading.ViciMultiposSelector.FlowSelector
    Bases: object
    __init__()
```

#### **Methods**

```
__init__()

getPort()

selectPort()
```

getPort()

selectPort()

### AFL.automation.loading.ViciMultiposSelector.SerialDevice

```
Bases: object
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

#### **Methods**

```
__init__(port[, baudrate, timeout, raw_writes])

sendCommand(cmd[, response, questionmarkOK,
...])
```

```
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

### AFL.automation.loading.ViciMultiposSelector.ViciMultiposSelector

Bases: SerialDevice, FlowSelector

\_\_init\_\_(port, baudrate=9600, portlabels=None)

connect to valve and query the number of positions

#### **Parameters**

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

#### **Methods**

```
__init__(port[, baudrate, portlabels]) connect to valve and query the number of positions

getPort([as_str]) query the current selected position

selectPort(port[, direction, block]) moves the selector to portnum

sendCommand(cmd[, response, questionmarkOK,
...])
```

\_\_init\_\_(port, baudrate=9600, portlabels=None)

connect to valve and query the number of positions

#### **Parameters**

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

selectPort(port, direction=None, block=True)

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

### **Parameters**

- port (String or int) the name, or number, of the port to switch to
- **direction** (*String*, *default=None*) direction "CW" or "CCW" to perform the move in
- **block** (bool, default=True) block return until move completes

```
getPort(as_str=False)
```

query the current selected position

sendCommand(cmd, response = True, questionmarkOK = False, timeout = -1, debug = False)

```
__init__(port, baudrate=9600, portlabels=None)
```

connect to valve and query the number of positions

#### **Parameters**

- to (port string describing the serial port the actuator is connected)
- use (baud baudrate to)
- naming (portlabels dict for smart port) 3,'instrument':4,'rinse':5,'waste':6}
- {'sample' (of the form) 3,'instrument':4,'rinse':5,'waste':6}

selectPort(port, direction=None, block=True)

moves the selector to portnum

if direction is set to either "CW" or "CCW" it moves the actuator in that direction. if unset or other value, will move via most efficient route.

#### **Parameters**

- port (String or int) the name, or number, of the port to switch to
- **direction** (*String*, *default=None*) direction "CW" or "CCW" to perform the move in
- block (bool, default=True) block return until move completes

getPort(as\_str=False)

query the current selected position

### 6.2.4 Prepare

The Prepare module provides functionality for sample preparation and processing.

AFL.automation.prepare

### AFL.automation.prepare

### **Functions**

```
##D20Factory(name[, phi_D2O, sld, properties]) Create a list of H2O/D2O solutions

compositionSweepFactory(name, components, ...)

make_locs(slot, nrows, ncols)

make_wellplate_locs(slot, size)
```

### AFL.automation.prepare.HD2OFactory

AFL.automation.prepare.HD20Factory(name, phi\_D2O=None, sld=None, properties=None)
Create a list of H2O/D2O solutions

### AFL.automation.prepare.compositionSweepFactory

AFL.automation.prepare.compositionSweepFactory(name, components, vary\_components, lo, hi, num, logspace=False, properties=None, progress=None)

### AFL.automation.prepare.make locs

AFL.automation.prepare.make\_locs(slot, nrows, ncols)

### AFL.automation.prepare.make\_wellplate\_locs

AFL.automation.prepare.make\_wellplate\_locs(slot, size)

### Classes

```
ComponentDB([path])

Deck()

MassBalance()

PipetteAction(source, dest, volume[, ...])

Sample(name, target[, target_check, balancer])

SampleSeries()

Solute(name[, description, density, ...]) Specialization class for solute components

Solution(name, components[, properties])

Solvent(name, density[, description, ...]) Specialization class for solute components
```

### AFL.automation.prepare.ComponentDB

```
class AFL.automation.prepare.ComponentDB(path='.afl/component.db.json')
    Bases: object
    __init__(path='.afl/component.db.json')
```

#### **Methods**

```
__init__([path])

add(name, preptype[, formula, density, sld, ...])

add_interactive(name)

read([path])

remove(name[, write])

write([path])

__init__(path='.afl/component.db.json')

read(path=None)

write(path=None)

add(name, preptype, formula=None, density=None, sld=None, write=False, description=None, overwrite=False)

add_interactive(name)

remove(name, write=False)
```

### AFL.automation.prepare.Deck

```
class AFL.automation.prepare.Deck
    Bases: object
    __init__()
```

### **Methods**

```
_init__()
 add_catch(name, slot)
 add_container(name, slot)
 add_pipette(name, mount, tipracks)
 add_stock(stock, location)
 add_target(target[, location, name])
 catch_sample(volume, source, dest[, ...])
 get_components()
 get_stock(name)
 init_remote_connection(url[, home])
 iterate_protocols()
 make_align_script(filename[,
 load_last_sample])
 make_mass_balance()
 make_protocol([only_validated, flatten, ...])
 make_sample_series([reset_sample_series])
 make_script(filename[, load_last_sample])
 reset_stocks()
 reset_targets()
 send_deck_config([debug_mode])
 send_protocol([send_deck_config, debug_mode])
 validate_sample_series([tolerance, ...])
__init__()
init_remote_connection(url, home=False)
catch_sample(volume, source, dest, mix_before=None, debug_mode=False)
send_deck_config(debug_mode=False)
```

```
send_protocol(send_deck_config=True, debug_mode=False)
     add_pipette(name, mount, tipracks)
     add_catch(name, slot)
     add_container(name, slot)
     add_stock(stock, location)
     get_stock(name)
     add_target(target, location='target', name=None)
     make_mass_balance()
     reset_targets()
     reset_stocks()
     get_components()
     make_sample_series(reset_sample_series=False)
     validate_sample_series(tolerance=0.0, print_report=True, progress=None)
     make_protocol(only_validated=False, flatten=False, pipette_kw=None)
     iterate_protocols()
     make_align_script(filename, load_last_sample=True)
     make_script(filename, load_last_sample=True)
AFL.automation.prepare.MassBalance
```

```
class AFL.automation.prepare.MassBalance
     Bases: object
     __init__()
```

### **Methods**

```
_init__()
 add_stock(stock, location)
 balance_mass()
 calculate_bounds([components, ...])
 constrain_samples_conc(constraints[, rtol])
 copy()
 in_bounds(points[, ternary])
 make_grid_mask([pts_per_row, ternary])
 make_mass_fraction_matrix()
 make_target_component_masses()
 plot_bounds([include_points])
 {\tt plot\_grid\_mask}()
 process_components()
 reset()
 reset\_stocks()
 reset_targets()
 sample_composition_space([pipette_min, ...])
                                                   Combine stock solutions to generate samples of pos-
                                                   sible target compositions
 set_target(target, location)
__init__()
copy()
add_stock(stock, location)
set_target(target, location)
reset_targets()
reset_stocks()
reset()
```

#### AFL.automation.prepare.PipetteAction

Bases: object

\_\_init\_\_(source, dest, volume, source\_loc=None, dest\_loc=None, aspirate\_rate=None, dispense\_rate=None, mix\_aspirate\_rate=None, mix\_dispense\_rate=None, mix\_before=None, mix\_after=None, blow\_out=False, post\_aspirate\_delay=0.0, post\_dispense\_delay=0.0, aspirate\_equilibration\_delay=0.0, drop\_tip=True, force\_new\_tip=False, to\_top=True, to\_center=False, to\_top\_z\_offset=0, fast\_mixing=False)

#### **Methods**

```
__init__(source, dest, volume[, source_loc, ...])
emit_protocol()
get_kwargs()
```

```
__init__(source, dest, volume, source_loc=None, dest_loc=None, aspirate_rate=None,
               dispense_rate=None, mix_aspirate_rate=None, mix_dispense_rate=None, mix_before=None,
               mix_after=None, blow_out=False, post_aspirate_delay=0.0, post_dispense_delay=0.0,
               aspirate\_equilibration\_delay=0.0, drop\_tip=True, force\_new\_tip=False, to\_top=True,
               to_center=False, to_top_z_offset=0, fast_mixing=False)
     emit_protocol()
     get_kwargs()
AFL.automation.prepare.Sample
class AFL.automation.prepare.Sample(name, target, target_check=None, balancer=None)
     Bases: object
     __init__(name, target, target_check=None, balancer=None)
     Methods
       __init__(name, target[, target_check, balancer])
      emit_protocol()
     Attributes
      balancer
      target
      target_check
      target_loc
     __init__(name, target, target_check=None, balancer=None)
     emit_protocol()
     property target
     property target_check
     property balancer
     property target_loc
```

### AFL.automation.prepare.SampleSeries

```
class AFL.automation.prepare.SampleSeries
    Bases: object
    __init__()
```

#### **Methods**

```
__init__()
add_sample(sample)

mass_totals_component([only_validated])

mass_totals_stock([only_validated])

reset()
shuffle()

reset()
```

```
reset()
shuffle()
add_sample(sample)
mass_totals_stock(only_validated=True)
```

mass\_totals\_component(only\_validated=True)

### AFL.automation.prepare.Solute

```
class AFL.automation.prepare.Solute(name, description=None, density=None, formula=None, sld=None)
    Bases: Component
    Specialization class for solute components
    __init__(name, description=None, density=None, formula=None, sld=None)
```

### **Methods**

```
__init__(name[, description, density, ...])

copy()

emit()

set_mass(value) Setter for inline mass changes
set_volume(volume) Setter for inline volume changes
```

### **Attributes**

```
density
 formula
 is_solute
 is_solvent
 mass
 moles
 sld
 volume
__init__(name, description=None, density=None, formula=None, sld=None)
property density
property formula
property sld
property volume
__hash__()
    Needed so Components can be dictionary keys
__iter__()
    Dummy iterator to mimic behavior of Mixture.
copy()
emit()
property is_solute
```

### AFL.automation.prepare.Solution

```
class AFL.automation.prepare.Solution(name, components, properties=None)
    Bases: object
    __init__(name, components, properties=None)
```

#### **Methods**

```
__init__(name, components[, properties])
add_component_from_name(name[,
contains(name)
copy([name])
from_dict(in_dict)
measure_out(amount[, deplete])
                                                  Create solution with identical composition at new to-
                                                  tal mass/volume
rename_component(old_name, new_name[,
place])
set_mass(value)
                                                  Setter for inline mass changes
set_properties_from_dict([properties,
                                             in-
place])
set_volume(value)
                                                  Setter for inline volume changes
to_dict()
```

#### **Attributes**

```
concentration
                                                Total mass of mixture.
mass
mass_fraction
                                                Mass fraction of components in mixture
molarity
size
solutes
solvent_density
solvent_mass
solvent_sld
solvent_volume
solvents
volume
                                                Total volume of mixture.
                                                Volume fraction of solvents in mixture
volume_fraction
```

```
__init__(name, components, properties=None)
__hash__()
    Needed so Solutions can be dictionary keys
to_dict()
classmethod from_dict(in_dict)
add_component_from_name(name, properties=None, inplace=False)
set_properties_from_dict(properties=None, inplace=False)
rename_component(old_name, new_name, inplace=False)
copy(name=None)
contains(name)
property size
property solutes
property solvents
__eq__(other)
     'Compare the mass, volume, and composition of two mixtures
property mass
     Total mass of mixture.
```

```
set_mass(value)
          Setter for inline mass changes
     property volume
          Total volume of mixture. Only solvents are included in volume calculation
     set_volume(value)
          Setter for inline volume changes
     property solvent_sld
     property solvent_density
     property solvent_volume
     property solvent_mass
     property mass_fraction
          Mass fraction of components in mixture
              Returns
                  • mass_fraction (dict)
                  • Component mass fractions
     property volume_fraction
          Volume fraction of solvents in mixture
              Returns
                  • solvent_fraction (dict)
                  • Component mass fractions
     property concentration
     property molarity
     measure_out(amount, deplete=False)
          Create solution with identical composition at new total mass/volume
AFL.automation.prepare.Solvent
class AFL.automation.prepare.Solvent(name, density, description=None, formula=None, sld=None)
     Bases: Component
     Specialization class for solute components
```

\_\_init\_\_(name, density, description=None, formula=None, sld=None)

# **Methods**

```
__init__(name, density[, description, ...])

copy()

emit()

set_mass(value) Setter for inline mass changes
set_volume(value) Setter for inline volume changes
```

# **Attributes**

```
density
 formula
 is_solute
 is_solvent
 mass
 moles
 sld
 volume
__init__(name, density, description=None, formula=None, sld=None)
property density
property formula
property sld
__hash__()
    Needed so Components can be dictionary keys
__iter__()
    Dummy iterator to mimic behavior of Mixture.
copy()
emit()
property is_solute
property is_solvent
```

```
property mass
property moles
set_mass(value)
    Setter for inline mass changes
set_volume(value)
    Setter for inline volume changes
property volume
```

# Modules

Component

DeckBuilderWidget

Dummy\_OT2\_Driver

OT2Client

PrepType

PrepareWidget

SampleSeriesWidget

StockBuilderWidget

SweepBuilderWidget

factory

utilities

# AFL.automation.prepare.Component

# **Functions**

enforce_units(value, unit_type)	Ensure that a number has units and convert to the de-
	fault_units

# AFL.automation.prepare.Component.enforce\_units

AFL.automation.prepare.Component.enforce\_units(value, unit\_type)

Ensure that a number has units and convert to the default\_units

#### **Classes**

Component(name, description)	Base class for all materials
<pre>PrepType(value[, names, module, qualname,])</pre>	

# AFL.automation.prepare.Component.Component

```
class AFL.automation.prepare.Component.Component(name, description)
```

Bases: object

Base class for all materials

This class defines all of the basic properties and methods to be shared across material objects

**\_\_init\_\_**(name, description)

### **Methods**

```
__init__(name, description)

copy()

emit()

set_mass(value) Setter for inline mass changes

set_volume(value) Setter for inline volume changes
```

# **Attributes**

```
density
 formula
 is_solute
 is_solvent
 mass
 moles
 sld
 volume
__init__(name, description)
emit()
__hash__()
    Needed so Components can be dictionary keys
copy()
__iter__()
    Dummy iterator to mimic behavior of Mixture.
property mass
set_mass(value)
    Setter for inline mass changes
property volume
set_volume(value)
    Setter for inline volume changes
property density
property formula
property moles
property sld
property is_solute
property is_solvent
```

# AFL.automation.prepare.Component.PrepType

```
__init__(*args, **kwds)
```

### **Attributes**

```
BaseComponent

BaseMixture

Solute

Solvent

Solution
```

```
BaseComponent = 1
BaseMixture = 2
Solute = 3
Solvent = 4
Solution = 5
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching *name*.

```
classmethod __iter__()
```

Return members in definition order.

# classmethod \_\_len\_\_()

Return the number of members (no aliases)

# **Exceptions**

ParseException(pstr[, loc, msg, elem])	Exception thrown when a parse expression doesn't match
	the input string

# AFL.automation.prepare.Component.ParseException

**exception** AFL.automation.prepare.Component.ParseException( $pstr: str, loc: int = 0, msg: str \mid None = None, elem=None)$ 

Exception thrown when a parse expression doesn't match the input string

Example:

```
integer = Word(nums).set_name("integer")
try:
   integer.parse_string("ABC")
except ParseException as pe:
   print(pe, f"column: {pe.column}")
```

```
prints:
     Expected integer, found 'ABC'
                                        (at char 0), (line:1, col:1) column: 1
     loc: int
     msg: str
     pstr: str
     parser_element: Any
     args: tuple[str, int, str | None]
class AFL.automation.prepare.Component.Component(name, description)
     Base class for all materials
     This class defines all of the basic properties and methods to be shared across material objects
     __init__(name, description)
     emit()
     __hash__()
          Needed so Components can be dictionary keys
     copy()
     __iter__()
          Dummy iterator to mimic behavior of Mixture.
```

property mass

set\_mass(value)

Setter for inline mass changes

```
property volume
set_volume(value)
    Setter for inline volume changes
property density
property formula
property moles
property sld
property is_solute
property is_solvent
```

# AFL.automation.prepare.DeckBuilderWidget

### **Functions**

sqrt(x, l)	Return the square root of x.	
------------	------------------------------	--

# AFL.automation.prepare.DeckBuilderWidget.sqrt

AFL.automation.prepare.DeckBuilderWidget. $\mathbf{sqrt}(x,/)$ Return the square root of x.

## **Classes**

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean <i>value</i> in the form of a checkbox.
<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
DeckBuilderWidget()	
<pre>DeckBuilderWidget_Model()</pre>	
<pre>DeckBuilderWidget_View()</pre>	
Dropdown(**kwargs)	Allows you to select a single item from a dropdown.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible
	box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible
	box model.

# AFL.automation.prepare.DeckBuilderWidget.Button

class AFL.automation.prepare.DeckBuilderWidget.Button(\*\*kwargs: Any)

Bases: DOMWidget, CoreWidget

Button widget.

This widget has an  $on\_click$  method that allows you to listen for the user clicking on the button. The click event itself is stateless.

### **Parameters**

- **description** (*str*) description displayed on the button
- icon (str) font-awesome icon names, without the 'fa-' prefix
- **disabled** (bool) whether user interaction is enabled

```
__init__(**kwargs)
```

Public constructor

### **Methods**

init(**kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
click()	Programmatically trigger a click event.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.

continues on next page

Table 17 – continued from previous page

	1 0
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_click(callback[, remove])</pre>	Register a callback to execute when the button is clicked.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
<pre>trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

### **Attributes**

button_style	Use a predefined styling for the button.
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Button label.
disabled	Enable or disable user changes.
icon	Font-awesome icon names, without the 'fa-' prefix.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

# description

Button label.

#### disabled

Enable or disable user changes.

#### icon

Font-awesome icon names, without the 'fa-' prefix.

#### button\_style

Use a predefined styling for the button.

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

```
__init__(**kwargs)
```

Public constructor

## on\_click(callback, remove=False)

Register a callback to execute when the button is clicked.

The callback will be called with one argument, the clicked button widget instance.

### **Parameters**

**remove** (bool (optional)) – Set to true to remove the callback from the list of callbacks.

# click()

Programmatically trigger a click event.

This will call the callbacks registered to the clicked button widget instance.

# \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow$ list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata:* Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

# classmethod close\_all()

#### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

#### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

#### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

#### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

#### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• **remove** (bool) – If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

# open()

Open a comm to the frontend if one isn't already open.

# remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

# tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str \mid None = None) \rightarrow dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

#### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = traitlets.All, type: traitlets.All, type:
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.DeckBuilderWidget.Checkbox

class AFL.automation.prepare.DeckBuilderWidget.Checkbox(\*\*kwargs: Any)

Bases: \_Bool

Displays a boolean value in the form of a checkbox.

#### **Parameters**

- value ({True, False}) value of the checkbox: True-checked, False-unchecked
- **description** (*str*) description displayed next to the checkbox
- **indent** ({*True*, *False*}) indent the control to align with other controls with a description. The style.description\_width attribute controls this width for consistence with other controls.

```
__init__(value=None, **kwargs)
Public constructor
```

## **Methods**

init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.

continues on next page

Table 18 – continued from previous page

ed from previous page
Return the value for this widget which should be
passed to interactive functions.
Returns the full state for a widget manager for embedding
Gets the widget state, or a piece of it.
Static method, called when a widget is constructed.
Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
Returns True if the object has a trait with the specified name.
Hold syncing any state until the outermost context manager exits
Context manager for bundling trait change notifications and cross validation.
Called when a property has changed.
Setup a handler to be called when a trait changes.
(Un)Register a custom msg receive callback.
DEPRECATED: Setup a handler to be called when a trait changes.
Registers a callback to be called when a widget is constructed.
Open a comm to the frontend if one isn't already open.
Removes a class from the top level element of the widget.
Sends a custom msg to the widget model in the frontend.
Sends the widget state, or a piece of it, to the front- end, if it exists.
Called when a state is received from the front-end.
Forcibly sets trait attribute, including read-only attributes.
This is called <b>before</b> selfinit is called.
Return a trait's default value or a dictionary of them
Get a dict of all the event handlers of this class.
Returns True if the specified trait has a value.
Get metadata values for trait by key.
Get a list of all the names of this class' traits.
A dict of trait names and their values.
Get a dict of all the traits of this class.
Remove a trait change handler.
Remove trait change handlers of any type for the specified name.

# **Attributes**

COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes.
indent	Indent the control to align with other controls with a description.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Bool value
widget_types	
widgets	

# indent

Indent the control to align with other controls with a description.

### style

Styling customizations

# \_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

# **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\*metadata: Any) $\rightarrow$ dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

#### **classmethod class\_trait\_names(\*\****metadata: Any*) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

#### classmethod close\_all()

## comm

A trait which allows any value.

#### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

#### description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated:: 8.0.0

Use tooltip attribute instead.

#### disabled

Enable or disable user changes.

#### focus()

Focus on the widget.

# get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

#### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

### get\_view\_spec()

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

#### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

#### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• **remove** (bool) – If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

# remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

# tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str \mid None = None) \rightarrow dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

#### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

**unobserve**(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = traitlets.All, type: traitlets.All, type:

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Bool value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.DeckBuilderWidget.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
```

# **Methods**

```
__init__([ip, port, username, interactive])

clear_history()

clear_queue()

debug(state)

deposit_obj(obj[, uid])

Deposit an object in the dropbox obj : object, the object to deposit id : str, the uuid to deposit the object under if not specified, a new uuid will be generated

driver_status()

enqueue([interactive])
```

continues on next page

Table 19 – continued from previous page

```
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get\_queued\_commands([inherit\_commands])
get_quickbar()
get_server_time()
get_unqueued_commmands([inherit_commands])
halt()
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
remove_item(uuid)
reset_queue_daemon()
retrieve_obj(uid[, delete])
                                                 Retrieve an object from the dropbox id: str, the uuid
                                                 of the object to retrieve delete: bool, if True, delete
                                                 the object after retrieving
server_cmd(cmd, **kwargs)
set_config([interactive])
set_driver_object(**kw)
set_object([serialize])
unqueued_base(**kwargs)
```

continues on next page

# Table 19 – continued from previous page

```
wait([target_uuid, interval, for_history, ...])
```

```
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
set_driver_object(**kw)
```

```
get_driver_object(name)

deposit_obj(obj, uid=None)

Deposit an object in the dropbox obj : object, the object to deposit id : str, the uuid to deposit the object under if not specified, a new uuid will be generated

retrieve_obj(uid, delete=True)

Retrieve an object from the dropbox id : str, the uuid of the object to retrieve delete : bool, if True, delete the object after retrieving

set_object(serialize=True, **kw)

get_object(name, serialize=True)
```

# AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget

### **Methods**

send\_deck\_cb(event)

```
__init__()
build_deck_object()

get_deck_config()

load_cb(event)

save_cb(event)

send_deck_cb(event)

set_deck_config(config)

start()

update_deck_graphic_cb(event)

__init__()

build_deck_object()

get_deck_config()

set_deck_config(config)
```

```
save_cb(event)
load_cb(event)
update_deck_graphic_cb(event)
start()
automation.prepare.DeckBuilde
```

# AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget Model

#### **Methods**

```
__init__()

build_deck_object(config)

send_deck_config([pi_ip, align_script])

__init__()

send_deck_config(pi_ip='piot2', align_script='/home/nistoroboto/align.py')

build_deck_object(config)
```

# AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget View

#### **Methods**

```
__init__()

create_expanded_button(description, button_style)
draw_deck()

start()
```

```
__init__()
create_expanded_button(description, button_style)
draw_deck()
start()
```

## AFL.automation.prepare.DeckBuilderWidget.Dropdown

```
class AFL.automation.prepare.DeckBuilderWidget.Dropdown(**kwargs: Any)
    Bases: _Selection
```

Allows you to select a single item from a dropdown.

### **Parameters**

- options (list) The options for the dropdown. This can either be a list of values, e.g. ['Galileo', 'Brahe', 'Hubble'] or [0, 1, 2], a list of (label, value) pairs, e.g. [('Galileo', 0), ('Brahe', 1), ('Hubble', 2)], or a Mapping between labels and values, e.g., {'Galileo': 0, 'Brahe': 1, 'Hubble': 2}.
- **index** (*int*) The index of the current selection.
- **value** (*any*) The value of the current selection. When programmatically setting the value, a reverse lookup is performed among the options to check that the value is valid. The reverse lookup uses the equality operator by default, but another predicate may be provided via the equals keyword argument. For example, when dealing with numpy arrays, one may set equals=np.array\_equal.
- **label** (*str*) The label corresponding to the selected value.
- **disabled** (bool) Whether to disable user changes.
- **description** (*str*) Label for this input group. This should be a string describing the widget.

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

## **Methods**

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	

continues on next page

422

Table 20 – continued from previous page

	ed from previous page
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
<pre>hold_trait_notifications()</pre>	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes
index	Selected index
keys	The traits which are synced.
label	Selected label
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
options	Iterable of values, (label, value) pairs, or Mapping be-
	tween labels and values that the user can select.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Selected value
widget_types	
widgets	

# \_\_del\_\_()

Object disposal

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

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

# classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# $\textbf{classmethod class\_own\_traits(**metadata: Any)} \rightarrow \text{dict[str, TraitType[Any, Any]]}$

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

#### classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

#### classmethod close\_all()

#### comm

A trait which allows any value.

#### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### description

Description of the control.

#### description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

## disabled

Enable or disable user changes

#### focus()

Focus on the widget.

### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

# $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### index

Selected index

#### kevs

The traits which are synced.

#### label

Selected label

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

## **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

# options

Iterable of values, (label, value) pairs, or Mapping between labels and values that the user can select.

The labels are the strings that will be displayed in the UI, representing the actual Python choices, and should be unique.

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

#### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

# **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

#### style

Styling customizations

#### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

# $trait_defaults(*names: str, **metadata: Any) \rightarrow dict[str, Any] | Sentinel$

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Selected value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.DeckBuilderWidget.HBox

Displays multiple widgets horizontally using the flexible box model.

## **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '.' Applies a predefined style to the box. Defaults to ', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Horizontal Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.HBox([title_widget, slider])
```

 $\verb"\__init__(children=(), **kwargs")$ 

Public constructor

# **Methods**

add_class(class(className)       Adds a class to the top level element of the widget.         add_traits(**traits)       Dynamically add trait attributes to the Widget.         blur()       Blur the widget.         class_own_trait_events(name)       Get a dict of all event handlers defined on this class not a parent.         class_trait_names(**metadata)       Get a dict of all the traitlets defined on this class, no a parent.         class_trait_names(**metadata)       Get a list of all the names of this class' traits.         close()       Close method.         close()       Close method.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler],	init([children])	Public constructor
add_traits(**traits)         Dynamically add trait attributes to the Widget.           blur()         Blur the widget.           class_own_trait_events(name)         Get a dict of all event handlers defined on this class not a parent.           class_own_traits(**metadata)         Get a dict of all the traitlets defined on this class, no a parent.           class_trait_names(**metadata)         Get a list of all the names of this class' traits.           close()         Close method.           close_all()         Focus on the widget.           focus()         Focus on the widget.           get_manager_state([drop_defaults, widgets])         Gets the widget state, or a piece of it.           get_view_spec()         Returns the full state for a widget manager for embedding           handle_comm_opened(comm, msg)         Static method, called when a widget is constructed.           handle_control_comm_opened(comm, msg)         Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received           has_trait(name)         Returns True if the object has a trait with the specified name.           hold_sync()         Hold syncing any state until the outermost contex manager exits           hold_trait_notifications()         Context manager for bundling trait change notifications and cross validation.           notify_change(change)         Setup a handler to be called when a trait changes.           on_		
blur()  class_own_trait_events(name)  class_own_traits(**metadata)  class_trait_names(**metadata)  class_traits(**metadata)  class_traits(**metadata)  close()  close_all()  focus()  get_manager_state([drop_defaults, widgets])  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  hold_sync()  hold_trait_notifications()  notify_change(change)  observe(handler, names, type])  on_trait_change([handler, name, remove])  open()  open()  open a comm to the frontend if one isn't already open remove_class(className)  et a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class.  Close method.  Close method.  Close method.  Close method.  Close method.  Clast weidget state, or a piece of it.   Static method, called when a widget is constructed.  Class method, called when a strait thange notification and cross validation.  Class method, called when a property has charged the parent of the decived on the parent of the parent of the parent of the decived of the parent of the decivery of the parent of the par		
class_owm_trait_events(name)       Get a dict of all event handlers defined on this class not a parent.         class_owm_traits(**metadata)       Get a dict of all the traitlets defined on this class, no a parent.         class_trait_names(**metadata)       Get a list of all the names of this class' traits.         close()       Close all (or all the traits of this class.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Focus on the widget state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Class method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       Cln/Register a custom msg receive callback.         on_widget_constructed(callback)       Registers a callback t	· · · · · · · · · · · · · · · · · · ·	•
not a parent.  Get a dict of all the traitlets defined on this class, no a parent.  Class_trait_names(**metadata)  Class_trait_s(**metadata)  Close()  Close adict of all the traitlets of this class' traits.  Close()  Close method.  Close method.  Close method.  Focus on the widget.  Returns the full state for a widget manager for embedding  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  Returns True if the object has a trait with the specified name.  hold_sync()  Hold syncing any state until the outermost contex manager exits  hold_trait_notifications()  notify_change(change)  observe(handler[, names, type])  on_msg(callback[, remove])  on_trait_change([handler, name, remove])  open()  open()  Open a comm to the frontend if one isn't already open remove_class(className)  Returns of all the traitlets defined on this class, no a parent.  Get a dict of all the traitlets defined on this class, no a parent.  Get a dict of all the traitlets defined on this class, no a parent.  Get a dict of all the traitlets defined on this class, no a parent.  Get a list of all the traitlets defined on this class, no a parent.  Get a list of all the traitlets defined on this class, no a parent.  Get a list of all the traitlets defined on this class' traits.  Class method.  Clase method.  Clase when a widget is constructed.  Class method, called when a vaidget is constructed.  Class method, called when a trait change notifications and cross validation.  Called when a property has changed.  Setup a handler to be called when a trait changes.  (Un)Register a custom msg receive callback.  Registers a callback to be called when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_class(className)	· · · · · · · · · · · · · · · · · · ·	ē
a parent.  Class_trait_names(**metadata)		not a parent.
class_traits(**metadata)       Get a dict of all the traits of this class.         close()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	class_own_traits(**metadata)	
close()       Close method.         close_all()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
Focus on the widget.   Gets manager_state([drop_defaults, widgets])   Returns the full state for a widget manager for embedding   Gets the widget state, or a piece of it.	class_traits(**metadata)	Get a dict of all the traits of this class.
focus()  get_manager_state([drop_defaults, widgets])  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  has_trait(name)  hold_sync()  hold_trait_notifications()  notify_change(change)  observe(handler[, names, type])  on_msg(callback[, remove])  on_widget_constructed(callback)  on_widget_constructed(callback)  open()  remove_class(className)  Focus on the widget.  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget is constructed.  Class method, called when a widget is constructed open ()  Popen a comm to the frontend if one isn't already open remove_class(className)  Returns the full state for a widget manager for embedding  Returns the full state for a widget is constructed.	close()	Close method.
get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)       Removes a class from the top level element of the	close_all()	
ding  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  has_trait(name)  hold_sync()  hold_trait_notifications()  conserve(handler[, names, type])  observe(handler[, names, type])  on_msg(callback[, remove])  on_widget_constructed(callback)  on_widget_constructed(callback)  remove_class(className)  ding  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Glass method, called when a widget is constructed.  Class method, called when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_clas	V .	<del>_</del>
get_view_spec()         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)       Removes a class from the top level element of the	<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open Remove_class(className)	<pre>get_view_spec()</pre>	
on the "jupyter.widget.control" comm channel is received  has_trait(name)  Returns True if the object has a trait with the specified name.  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  Observe(handler[, names, type])  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open remove_class(className)  Removes a class from the top level element of the	<pre>handle_comm_opened(comm, msg)</pre>	
name.  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open Remove_class(className)  Removes a class from the top level element of the	handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  Om_msg(callback[, remove])  Called when a property has changed.  Setup a handler to be called when a trait changes.  (Un)Register a custom msg receive callback.  Om_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  Om_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open remove_class(className)  Removes a class from the top level element of the	has_trait(name)	Returns True if the object has a trait with the specified name.
hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  On_widget_constructed(callback)  Open()  Open a comm to the frontend if one isn't already open Remove_class(className)  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Open a common to the frontend if one isn't already open notifications and cross validation.	hold_sync()	Hold syncing any state until the outermost context manager exits
observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the	hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the	notify_change(change)	Called when a property has changed.
on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		
on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		
on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		DEPRECATED: Setup a handler to be called when a
remove_class(className) Removes a class from the top level element of the	on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
	open()	Open a comm to the frontend if one isn't already open.
widget.	remove_class(className)	Removes a class from the top level element of the widget.

continues on next page

Table 21 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

### **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
    Object disposal
__init__(children=(), **kwargs)
    Public constructor

add_class(className)
    Adds a class to the top level element of the widget.
    Doesn't add the class if it already exists.

add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

### box\_style

Use a predefined styling for the box.

#### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

### classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

432

Focus on the widget.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

#### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

# tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

## **Parameters**

**name** (*str* (*default: None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

# Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

## **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.DeckBuilderWidget.Label

```
class AFL.automation.prepare.DeckBuilderWidget.Label(**kwargs: Any)
```

Bases: \_String

Label widget.

It also renders math inside the string *value* as Latex (requires \$ \$ or \$\$ \$\$ and similar latex tags).

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

### **Methods**

init ([l])	D.hl: a construction
init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
	continues on next nage

continues on next page

Table 22 – continued from previous page

	a nom previous page
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been
	typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

### style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### \_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

# classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

### **classmethod class\_trait\_names(\*\****metadata: Any*) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

#### classmethod close\_all()

### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

#### focus()

Focus on the widget.

#### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

# layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

# placeholder

Placeholder text to display when nothing has been typed

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

#### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

# $trait_defaults(*names: str, **metadata: Any) \rightarrow dict[str, Any] | Sentinel$

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.DeckBuilderWidget.Layout

```
class AFL.automation.prepare.DeckBuilderWidget.Layout(**kwargs: Any)
```

Bases: Widget

Layout specification

Defines a layout that can be expressed using CSS. Supports a subset of https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

When a property is also accessible via a shorthand property, we only expose the shorthand.

For example: - flex-grow, flex-shrink and flex-basis are bound to flex. - flex-wrap and flex-direction are bound to flex-flow. - margin-[top/bottom/left/right] values are bound to margin, etc.

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

Public constructor

# Methods

I I A Calculat	D.11
init(**kwargs)	Public constructor
add_traits(**traits)	Dynamically add trait attributes to the Widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
	continues on poyt page

continues on next page

Table 23 – continued from previous page

trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

7.4	The state of the s
align_content	The align-content CSS attribute.
align_items	The align-items CSS attribute.
align_self	The align-self CSS attribute.
border	border property getter.
border_bottom	The border bottom CSS attribute.
border_left	The border left CSS attribute.
border_right	The border right CSS attribute.
border_top	The border top CSS attribute.
bottom	The bottom CSS attribute.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
display	The display CSS attribute.
flex	The flex CSS attribute.
flex_flow	The flex-flow CSS attribute.
grid_area	The grid-area CSS attribute.
grid_auto_columns	The grid-auto-columns CSS attribute.
grid_auto_flow	The grid-auto-flow CSS attribute.
grid_auto_rows	The grid-auto-rows CSS attribute.
grid_column	The grid-column CSS attribute.
grid_gap	The grid-gap CSS attribute.
grid_row	The grid-row CSS attribute.
grid_template_areas	The grid-template-areas CSS attribute.
grid_template_columns	The grid-template-columns CSS attribute.
grid_template_rows	The grid-template-rows CSS attribute.
height	The height CSS attribute.
justify_content	The justify-content CSS attribute.
justify_items	The justify-items CSS attribute.
keys	The traits which are synced.
left	The left CSS attribute.
log	A trait whose value must be an instance of a specified
	class.
margin	The margin CSS attribute.
max_height	The max-height CSS attribute.
max_width	The max-width CSS attribute.
min_height	The min-height CSS attribute.
min_width	The min-width CSS attribute.
model_id	Gets the model id of this widget.
object_fit	The object-fit CSS attribute.
object_position	The object-position CSS attribute.
order	The order CSS attribute.
	continues on next page

continues on next page

Table 24 – continued from previous page

overflow	The overflow CSS attribute.
padding	The padding CSS attribute.
right	The right CSS attribute.
top	The top CSS attribute.
visibility	The visibility CSS attribute.
widget_types	
widgets	
width	The width CSS attribute.

# align\_content

The align-content CSS attribute.

# align\_items

The align-items CSS attribute.

# align\_self

The align-self CSS attribute.

# border\_top

The border top CSS attribute.

# border\_right

The border right CSS attribute.

### border\_bottom

The border bottom CSS attribute.

# border\_left

The border left CSS attribute.

# bottom

The bottom CSS attribute.

### display

The display CSS attribute.

# flex

The flex CSS attribute.

# flex\_flow

The flex-flow CSS attribute.

# height

The height CSS attribute.

# justify\_content

The justify-content CSS attribute.

# justify\_items

The justify-items CSS attribute.

### left

The left CSS attribute.

### margin

The margin CSS attribute.

# max\_height

The max-height CSS attribute.

# max\_width

The max-width CSS attribute.

# min\_height

The min-height CSS attribute.

# min\_width

The min-width CSS attribute.

### overflow

The overflow CSS attribute.

# order

The order CSS attribute.

# padding

The padding CSS attribute.

# right

The right CSS attribute.

#### top

The top CSS attribute.

# visibility

The visibility CSS attribute.

# width

The width CSS attribute.

# object\_fit

The object-fit CSS attribute.

# object\_position

The object-position CSS attribute.

# grid\_auto\_columns

The grid-auto-columns CSS attribute.

### grid\_auto\_flow

The grid-auto-flow CSS attribute.

# grid\_auto\_rows

The grid-auto-rows CSS attribute.

### grid\_gap

The grid-gap CSS attribute.

# grid\_template\_rows

The grid-template-rows CSS attribute.

#### grid\_template\_columns

The grid-template-columns CSS attribute.

## grid\_template\_areas

The grid-template-areas CSS attribute.

### grid\_row

The grid-row CSS attribute.

### grid\_column

The grid-column CSS attribute.

### grid\_area

The grid-area CSS attribute.

```
__init__(**kwargs)
```

Public constructor

### property border

border property getter. Return the common value of all side borders if they are identical. Otherwise return None.

```
__del__()
```

Object disposal

```
add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

## **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

```
classmethod class_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### Returns

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

#### get\_view\_spec()

```
static handle_comm_opened(comm, msg)
```

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

# hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

# property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

### on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

# **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

### **Parameters**

**key** (unicode, or iterable (optional)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait\_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) \rightarrow dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# **Return type**

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.DeckBuilderWidget.Text

# **Methods**

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	

continues on next page

Table 25 – continued from previous page

<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
<pre>handle_control_comm_opened(comm, msg)</pre>	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica-
	tions and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_submit(callback[, remove])</pre>	(Un)Register a callback to handle text submission.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a
	trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is con-
	structed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the
	widget.
<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the front-
	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
	end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only at-
	tributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
<pre>trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
<pre>unobserve_all([name])</pre>	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

comm	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

# disabled

Enable or disable user changes

# continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

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

Public constructor

# on\_submit(callback, remove=False)

(Un)Register a callback to handle text submission.

Triggered when the user clicks enter.

# **Parameters**

- callback (callable) Will be called with exactly one argument: the Widget instance
- remove (bool (optional)) Whether to unregister the callback

# \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# **classmethod class\_own\_traits**(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow list[str]$

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

## comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

#### focus()

Focus on the widget.

### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

 $\textbf{key} \, (unicode \ or \ iterable \ (optional)) - A \ single \ property's \ name \ or \ iterable \ of \ property \ names \ to \ get.$ 

#### **Returns**

- **state** (dict of states)
- metadata (dict) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

#### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

#### placeholder

Placeholder text to display when nothing has been typed

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

# send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (list of binary buffers) Binary buffers to send with message

# send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

#### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

#### setup\_instance(\*\*kwargs: Any) $\rightarrow$ None

This is called **before** self.\_\_init\_\_ is called.

# tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.DeckBuilderWidget.VBox

Displays multiple widgets vertically using the flexible box model.

### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '.' Applies a predefined style to the box. Defaults to ', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Vertical Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.VBox([title_widget, slider])
```

 $\verb|\__init__(children=(), **kwargs)|$ 

Public constructor

# **Methods**

init([children])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class,
` '	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	, 1
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
has_trait(name)	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica-
	tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a
7	trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is con-
. , ,	structed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the
	widget.
	continues on next nage

continues on next page

Table 26 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
   Object disposal
__init__(children=(), **kwargs)
   Public constructor

add_class(className)
   Adds a class to the top level element of the widget.
   Doesn't add the class if it already exists.

add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

### blur()

Blur the widget.

### box\_style

Use a predefined styling for the box.

### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

# $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

# keys

The traits which are synced.

#### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

# property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### Parameters

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

# **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

# **Parameters**

**name** (*str* (*default: None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

# Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
     widgets = {}
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget
     __init__()
     build_deck_object()
     get_deck_config()
     set_deck_config(config)
     send_deck_cb(event)
     save_cb(event)
     load_cb(event)
     update_deck_graphic_cb(event)
     start()
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_Model
     __init__()
     send_deck_config(pi_ip='piot2', align_script='/home/nistoroboto/align.py')
     build_deck_object(config)
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_View
     __init__()
     create_expanded_button(description, button_style)
     draw_deck()
     start()
```

# AFL.automation.prepare.Dummy\_OT2\_Driver

# Classes

Driver(name[, defaults, overrides])

Dummy\_OT2\_Driver([overrides])

# AFL.automation.prepare.Dummy\_OT2\_Driver.Driver

class AFL.automation.prepare.Dummy\_OT2\_Driver.Driver(name, defaults=None, overrides=None)
 Bases: object
 \_\_init\_\_(name, defaults=None, overrides=None)

## **Methods**

```
_init__(name[, defaults, overrides])
                                                    Store an object in the dropbox
 deposit_obj(obj[, uid])
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

• uid(str) – The uuid to store the object under

# AFL.automation.prepare.Dummy\_OT2\_Driver.Dummy\_OT2\_Driver

```
class AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driver(overrides=None)
    Bases: Driver
    __init__(overrides=None)
```

### **Methods**

```
_init__([overrides])
add_prep_targets(targets[, reset])
deactivate_temp(slot)
                                                   Disablea tempdeck in slot slot
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_labware(slot)
get_object(name[, serialize])
get_prep_target()
get_sample()
get_shake_latch_status()
get_shake_rpm()
get_shaker_temp()
                                                   Get the temperature of a tempdeck in slot slot
get_temp(slot)
get_wells(locs)
home(**kwargs)
latch_shaker()
load_instrument(name, mount, tip_rack_slots, ...)
load_labware(name, slot[, module])
parse_well(loc)
post_execute(**kwargs)
                                                  Executed after each call to execute
```

continues on next page

Table 27 – continued from previous page

```
pre_execute(**kwargs)
                                                  Executed before each call to execute
queued()
quickbar()
reset_prep_targets()
reset_sample()
reset_tipracks([mount])
retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox
set_aspirate_rate([rate])
set_config(**kwargs)
set_data(data)
                                                  Set data in the DataPacket object
set_dispense_rate([rate])
set_gantry_speed([speed])
set_object([serialized])
set_sample(sample_name[, sample_uuid])
set_shake(rpm)
set_shaker_temp(temp)
set_temp(slot, temp)
                                                  Set the temperature of a tempdeck in slot slot
status()
stop_shake()
transfer(source, dest, volume, *args, **kwargs)
unlatch_shaker()
unqueued()
```

# **Attributes**

```
defaults

defaults = {'execute_delay': 1}
__init__(overrides=None)
```

```
reset_prep_targets()
add_prep_targets(targets, reset=False)
get_prep_target()
status()
reset_tipracks(mount='both')
home(**kwargs)
parse_well(loc)
get_wells(locs)
get_labware(slot)
load_labware(name, slot, module=None, **kwargs)
set_temp(slot, temp)
    Set the temperature of a tempdeck in slot slot
get_temp(slot)
    Get the temperature of a tempdeck in slot slot
deactivate_temp(slot)
    Disablea tempdeck in slot slot
set_shake(rpm)
stop_shake()
set_shaker_temp(temp)
unlatch_shaker()
latch_shaker()
get_shaker_temp()
get_shake_rpm()
get_shake_latch_status()
load_instrument(name, mount, tip_rack_slots, **kwargs)
transfer(source, dest, volume, *args, **kwargs)
set_aspirate_rate(rate=150)
set_dispense_rate(rate=300)
set_gantry_speed(speed=400)
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
```

- **obj** (*object*) The object to store in the dropbox
- **uid** (*str*) The uuid to store the object under

```
execute(**kwargs)
     classmethod gather_defaults()
          Gather all inherited static class-level dictionaries called default.
     get_config(name, print_console=False)
     get_configs(print_console=False)
     get_object(name, serialize=True)
     get_sample()
     post_execute(**kwargs)
          Executed after each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     pre_execute(**kwargs)
          Executed before each call to execute
          All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
          by subclasses.
     queued()
     quickbar()
     reset_sample()
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
               Parameters
                  uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
               Parameters
                   • data (dict) – Dictionary of data to store in the driver object
                   • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
class AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driver(overrides=None)
     defaults = {'execute_delay': 1}
     __init__(overrides=None)
```

```
reset_prep_targets()
add_prep_targets(targets, reset=False)
get_prep_target()
status()
reset_tipracks(mount='both')
home(**kwargs)
parse_well(loc)
get_wells(locs)
get_labware(slot)
load_labware(name, slot, module=None, **kwargs)
set_temp(slot, temp)
     Set the temperature of a tempdeck in slot slot
get_temp(slot)
    Get the temperature of a tempdeck in slot slot
deactivate_temp(slot)
    Disablea tempdeck in slot slot
set_shake(rpm)
stop_shake()
set_shaker_temp(temp)
unlatch_shaker()
latch_shaker()
get_shaker_temp()
get_shake_rpm()
get_shake_latch_status()
load_instrument(name, mount, tip_rack_slots, **kwargs)
transfer(source, dest, volume, *args, **kwargs)
set_aspirate_rate(rate=150)
set_dispense_rate(rate=300)
set_gantry_speed(speed=400)
```

# AFL.automation.prepare.OT2Client

## **Classes**

<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
<pre>OT2Client([ip, port, username, interactive])</pre>	Communicate with AFL-automation server on OT-2
<pre>PipetteAction(source, dest, volume[,])</pre>	

# AFL.automation.prepare.OT2Client.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

**\_\_init\_\_**(*ip=None*, *port='5000'*, *username=None*, *interactive=False*)

# **Methods**

init([ip, port, username, interactive])	
<pre>clear_history()</pre>	
clear_queue()	
debug(state)	
<pre>deposit_obj(obj[, uid])</pre>	Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object under if not specified, a new uuid will be generated
<pre>driver_status()</pre>	
enqueue([interactive])	
enqueued_base(**kwargs)	
<pre>from_server_name(server_name, **kwargs)</pre>	
<pre>get_config(name[, print_console, interactive])</pre>	
<pre>get_driver_object(name)</pre>	
<pre>get_object(name[, serialize])</pre>	

continues on next page

Table 28 – continued from previous page

```
get_queue()
 get_queued_commands([inherit_commands])
 get_quickbar()
 get_server_time()
 get_unqueued_commands([inherit_commands])
 halt()
 logged_in()
 login(username[, populate_commands])
 move_item(uuid, pos)
 pause(state)
 query_driver(**kwargs)
 queue_state()
 remove_item(uuid)
 reset_queue_daemon()
 retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox id: str, the uuid
                                                  of the object to retrieve delete: bool, if True, delete
                                                  the object after retrieving
 server_cmd(cmd, **kwargs)
 set_config([interactive])
 set_driver_object(**kw)
 set_object([serialize])
 unqueued_base(**kwargs)
 wait([target_uuid, interval, for_history, ...])
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
```

```
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
set_driver_object(**kw)
get_driver_object(name)
deposit_obj(obj, uid=None)
     Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
     under if not specified, a new uuid will be generated
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
     the object after retrieving
set_object(serialize=True, **kw)
get_object(name, serialize=True)
```

# AFL.automation.prepare.OT2Client.OT2Client

Bases: Client

Communicate with AFL-automation server on OT-2

This class maps pipettor functions to HTTP REST requests that are sent to the AFL-automation server

**\_\_init\_\_**(*ip=None*, *port='5000'*, *username=None*, *interactive=False*)

# **Methods**

```
_init__([ip, port, username, interactive])
aspirate_rate(rate)
clear_history()
clear_queue()
debug(state)
deposit_obj(obj[, uid])
                                                   Deposit an object in the dropbox obj: object, the ob-
                                                  ject to deposit id: str, the uuid to deposit the object
                                                   under if not specified, a new uuid will be generated
dispense_rate(rate)
driver_status()
enqueue([interactive])
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
```

continues on next page

Table 29 - continued from previous page

```
get_unqueued_commmands([inherit_commands])
halt()
home()
load_instrument(name, mount, tip_rack_slots)
load_labware(name, slot)
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
remove_item(uuid)
reset_queue_daemon()
reset_tipracks([mount])
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox id: str, the uuid
                                                   of the object to retrieve delete: bool, if True, delete
                                                   the object after retrieving
server_cmd(cmd, **kwargs)
set_config([interactive])
set_driver_object(**kw)
set_object([serialize])
                                                   Transfer fluid from one location to another
transfer(source, dest, volume[, interactive])
unqueued_base(**kwargs)
wait([target_uuid, interval, for_history, ...])
```

transfer(source, dest, volume, interactive=None, \*\*kwargs)

Transfer fluid from one location to another

#### **Parameters**

• source (str or list of str Source wells to transfer from. Wells should be specified as three) — character strings with the first character being the slot number.

```
    dest (str or list of str) – Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
    volume (float) – volume of fluid to transfer in microliters
```

```
reset_tipracks(mount='both')
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
aspirate_rate(rate)
dispense_rate(rate)
home()
__init__(ip=None, port='5000', username=None, interactive=False)
clear_history()
clear_queue()
debug(state)
deposit_obj(obj, uid=None)
    Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
    under if not specified, a new uuid will be generated
driver_status()
enqueue(interactive=None, **kwargs)
enqueued_base(**kwargs)
classmethod from_server_name(server_name, **kwargs)
get_config(name, print_console=True, interactive=None)
get_driver_object(name)
get_object(name, serialize=True)
get_queue()
get_queued_commands(inherit_commands=True)
get_quickbar()
get_server_time()
get_unqueued_commands(inherit_commands=True)
halt()
logged_in()
login(username, populate_commands=True)
move_item(uuid, pos)
```

```
pause(state)
query_driver(**kwargs)
queue_state()
remove_item(uuid)
reset_queue_daemon()
retrieve_obj(uid, delete=True)
    Retrieve an object from the dropbox id : str, the uuid of the object to retrieve delete : bool, if True, delete the object after retrieving
server_cmd(cmd, **kwargs)
set_config(interactive=None, **kwargs)
set_driver_object(**kw)
set_object(serialize=True, **kw)
unqueued_base(**kwargs)
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
```

# AFL.automation.prepare.OT2Client.PipetteAction

```
class AFL.automation.prepare.OT2Client.PipetteAction(source, dest, volume, source_loc=None, dest_loc=None, aspirate_rate=None, dispense_rate=None, mix_aspirate_rate=None, mix_dispense_rate=None, mix_before=None, mix_after=None, blow_out=False, post_aspirate_delay=0.0, post_dispense_delay=0.0, aspirate_equilibration_delay=0.0, drop_tip=True, force_new_tip=False, to_top=True, to_center=False, to_top_z_offset=0, fast_mixing=False)
```

Bases: object

\_\_init\_\_(source, dest, volume, source\_loc=None, dest\_loc=None, aspirate\_rate=None, dispense\_rate=None, mix\_aspirate\_rate=None, mix\_dispense\_rate=None, mix\_before=None, mix\_after=None, blow\_out=False, post\_aspirate\_delay=0.0, post\_dispense\_delay=0.0, aspirate\_equilibration\_delay=0.0, drop\_tip=True, force\_new\_tip=False, to\_top=True, to\_center=False, to\_top\_z\_offset=0, fast\_mixing=False)

## **Methods**

```
__init__(source, dest, volume[, source_loc, ...])

emit_protocol()

get_kwargs()
```

\_\_init\_\_(source, dest, volume, source\_loc=None, dest\_loc=None, aspirate\_rate=None, dispense\_rate=None, mix\_aspirate\_rate=None, mix\_dispense\_rate=None, mix\_before=None, mix\_after=None, blow\_out=False, post\_aspirate\_delay=0.0, post\_dispense\_delay=0.0, aspirate\_equilibration\_delay=0.0, drop\_tip=True, force\_new\_tip=False, to\_top=True, to\_center=False, to\_top\_z\_offset=0, fast\_mixing=False)

```
emit_protocol()
get_kwargs()
```

Communicate with AFL-automation server on OT-2

This class maps pipettor functions to HTTP REST requests that are sent to the AFL-automation server

transfer(source, dest, volume, interactive=None, \*\*kwargs)

Transfer fluid from one location to another

#### **Parameters**

- source (str or list of str Source wells to transfer from. Wells should be specified as three) character strings with the first character being the slot number.
- **dest**(*str or list of str*) Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **volume** (*float*) volume of fluid to transfer in microliters

```
reset_tipracks(mount='both')
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
aspirate_rate(rate)
dispense_rate(rate)
home()
```

# AFL.automation.prepare.PrepType

### **Functions**

```
makeRegistar()

prepRegistrar(prepType)
```

# AFL.automation.prepare.PrepType.makeRegistar

AFL.automation.prepare.PrepType.makeRegistar()

# AFL.automation.prepare.PrepType.prepRegistrar

AFL.automation.prepare.PrepType.prepRegistrar(prepType)

## Classes

Enum(value[, names, module, qualname, type,])	Create a collection of name/value pairs.
<pre>PrepType(value[, names, module, qualname,])</pre>	
auto([value])	Instances are replaced with an appropriate value in Enum class suites.

# AFL.automation.prepare.PrepType.Enum

Bases: object

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
... RED = 1
... BLUE = 2
... GREEN = 3
```

Access them by:

• attribute access:

```
>>> Color.RED <Color.RED: 1>
```

• value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

• name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

```
__init__(*args, **kwds)
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching name.

```
classmethod __iter__()
```

Return members in definition order.

```
classmethod __len__()
```

Return the number of members (no aliases)

# AFL.automation.prepare.PrepType.PrepType

```
__init__(*args, **kwds)
```

## **Attributes**

```
BaseMixture

Solute

Solvent

Solution
```

```
BaseComponent = 1
BaseMixture = 2
Solute = 3
Solvent = 4
Solution = 5
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)

Return the member matching name.

classmethod __iter__()

Return members in definition order.
```

classmethod \_\_len\_\_()

Return the number of members (no aliases)

# AFL.automation.prepare.PrepType.auto

```
class AFL.automation.prepare.PrepType.auto(value=_auto_null)
    Bases: object
    Instances are replaced with an appropriate value in Enum class suites.
    __init__(value=_auto_null)
```

# **Methods**

```
__init__([value])

__init__(value=_auto_null)

class AFL.automation.prepare.PrepType.PrepType(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)

BaseComponent = 1

BaseMixture = 2

Solute = 3

Solvent = 4

Solution = 5
```

# AFL.automation.prepare.PrepareWidget

## **Functions**

sqrt(x, l) Return the square root of x.

# AFL.automation.prepare.PrepareWidget.sqrt

AFL.automation.prepare.PrepareWidget. $\mathbf{sqrt}(x,/)$ Return the square root of x.

# Classes

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean <i>value</i> in the form of a checkbox.
DeckBuilderWidget()	
Dropdown(**kwargs)	Allows you to select a single item from a dropdown.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
PrepareWidget()	
PrepareWidget_Model()	
<pre>PrepareWidget_View()</pre>	
SampleSeriesWidget(deck)	
StockBuilderWidget(deck)	
SweepBuilderWidget(deck)	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible box model.

# AFL.automation.prepare.PrepareWidget.Button

class AFL.automation.prepare.PrepareWidget.Button(\*\*kwargs: Any)

Bases: DOMWidget, CoreWidget

Button widget.

This widget has an  $on\_click$  method that allows you to listen for the user clicking on the button. The click event itself is stateless.

### **Parameters**

- description(str) description displayed on the button
- icon (str) font-awesome icon names, without the 'fa-' prefix
- **disabled** (bool) whether user interaction is enabled

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

Public constructor

# Methods

ini+ (**1)	Dalatia accordantes
init(**kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
click()	Programmatically trigger a click event.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	, 1
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica- tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_click(callback[, remove])	Register a callback to execute when the button is clicked.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
	continues on next page

continues on next page

Table 30 – continued from previous page

<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
<pre>trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

button_style	Use a predefined styling for the button.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Button label.
disabled	Enable or disable user changes.
icon	Font-awesome icon names, without the 'fa-' prefix.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

# description

Button label.

## disabled

Enable or disable user changes.

## icon

Font-awesome icon names, without the 'fa-' prefix.

# button\_style

Use a predefined styling for the button.

# style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# \_\_init\_\_(\*\*kwargs)

Public constructor

### on\_click(callback, remove=False)

Register a callback to execute when the button is clicked.

The callback will be called with one argument, the clicked button widget instance.

#### **Parameters**

**remove** (bool (optional)) – Set to true to remove the callback from the list of callbacks.

### click()

Programmatically trigger a click event.

This will call the callbacks registered to the clicked button widget instance.

# \_\_del\_\_()

Object disposal

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

### blur()

Blur the widget.

## **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### **classmethod class\_own\_traits**(\*\*metadata: Any) $\rightarrow$ dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

```
classmethod class_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

# get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

# **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

# static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

# hold\_sync()

Hold syncing any state until the outermost context manager exits

### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

# property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

# notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

# **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

on\_trait\_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False)  $\rightarrow None$ 

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

# send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (list of binary buffers) Binary buffers to send with message

# send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

# set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

# $setup_instance(**kwargs: Any) \rightarrow None$

This is called **before** self.\_\_init\_\_ is called.

### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.PrepareWidget.Checkbox

```
class AFL.automation.prepare.PrepareWidget.Checkbox(**kwargs: Any)
```

Bases: \_Bool

Displays a boolean value in the form of a checkbox.

### **Parameters**

- value ({True, False}) value of the checkbox: True-checked, False-unchecked
- **description** (*str*) description displayed next to the checkbox
- **indent** ({*True*, *False*}) indent the control to align with other controls with a description. The style description width attribute controls this width for consistence with other controls.

```
__init__(value=None, **kwargs)
```

Public constructor

# Methods

/ / (F 1 1)	D 11:
init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
<b>V</b>	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	•
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re-
	ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
<pre>observe(handler[, names, type])</pre>	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
	continues on next page

continues on next page

Table 31 – continued from previous page

<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes.
indent	Indent the control to align with other controls with a description.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Bool value
widget_types	
widgets	

## indent

Indent the control to align with other controls with a description.

# style

Styling customizations

\_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

## add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

## **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## **classmethod class\_trait\_names(\*\****metadata: Any*) → list[str]

Get a list of all the names of this class' traits.

This method is just like the *trait\_names()* method, but is unbound.

# **classmethod class\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

## description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

#### disabled

Enable or disable user changes.

#### focus()

Focus on the widget.

## get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

## **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

## Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### **Returns**

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (*list of binary buffers*) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

## tabbable

Is widget tabbable?

## tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Bool value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.PrepareWidget.DeckBuilderWidget

## **Methods**

```
__init__()

build_deck_object()

get_deck_config()

load_cb(event)

save_cb(event)

send_deck_cb(event)

set_deck_config(config)

start()

update_deck_graphic_cb(event)
```

```
__init__()
build_deck_object()
get_deck_config()
set_deck_config(config)
send_deck_cb(event)
save_cb(event)
load_cb(event)
update_deck_graphic_cb(event)
start()
```

## AFL.automation.prepare.PrepareWidget.Dropdown

Allows you to select a single item from a dropdown.

## **Parameters**

- **options** (*list*) The options for the dropdown. This can either be a list of values, e.g. ['Galileo', 'Brahe', 'Hubble'] or [0, 1, 2], a list of (label, value) pairs, e.g. [('Galileo', 0), ('Brahe', 1), ('Hubble', 2)], or a Mapping between labels and values, e.g., {'Galileo': 0, 'Brahe': 1, 'Hubble': 2}.
- **index** (*int*) The index of the current selection.

- **value** (*any*) The value of the current selection. When programmatically setting the value, a reverse lookup is performed among the options to check that the value is valid. The reverse lookup uses the equality operator by default, but another predicate may be provided via the equals keyword argument. For example, when dealing with numpy arrays, one may set equals=np.array\_equal.
- **label** (*str*) The label corresponding to the selected value.
- **disabled** (*bool*) Whether to disable user changes.
- **description** (*str*) Label for this input group. This should be a string describing the widget.

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

### **Methods**

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be
	passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.

continues on next page

Table 32 – continued from previous page

on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a
on_ on the first of the first o	trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
<pre>trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes
index	Selected index
keys	The traits which are synced.
label	Selected label
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
options	Iterable of values, (label, value) pairs, or Mapping be-
	tween labels and values that the user can select.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Selected value
widget_types	
widgets	

\_\_del\_\_()

Object disposal

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

Public constructor

## add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

## add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

# classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# $\textbf{classmethod class\_own\_traits}(**metadata: Any) \rightarrow \text{dict}[\text{str}, \text{TraitType}[\text{Any}, \text{Any}]]$

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow$ list[str]

Get a list of all the names of this class' traits.

This method is just like the *trait\_names()* method, but is unbound.

### classmethod class\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

### description

Description of the control.

#### description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

## disabled

Enable or disable user changes

# focus()

Focus on the widget.

## get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## index

Selected index

#### kevs

The traits which are synced.

#### label

Selected label

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

## options

Iterable of values, (label, value) pairs, or Mapping between labels and values that the user can select.

The labels are the strings that will be displayed in the UI, representing the actual Python choices, and should be unique.

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

#### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

## **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

#### style

Styling customizations

#### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

# $trait_defaults(*names: str, **metadata: Any) \rightarrow dict[str, Any] | Sentinel$

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Selected value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.PrepareWidget.HBox

Displays multiple widgets horizontally using the flexible box model.

### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '. Applies a predefined style to the box. Defaults to '', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Horizontal Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.HBox([title_widget, slider])
```

```
__init__(children=(), **kwargs)
Public constructor
```

## **Methods**

init([children])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a
	trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.

continues on next page

Table 33 - continued from previous page

	a nem previede page
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
```

Object disposal

\_\_init\_\_(children=(), \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

## box\_style

Use a predefined styling for the box.

#### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

# layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default:* '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

## tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
\textbf{classmethod trait\_events}(\textit{name: str} \mid \textit{None} = \textit{None}) \rightarrow \text{dict}[\textit{str}, \textit{EventHandler}]
```

Get a dict of all the event handlers of this class.

## **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

## **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.PrepareWidget.Label

```
class AFL.automation.prepare.PrepareWidget.Label(**kwargs: Any)
```

Bases: \_String

Label widget.

It also renders math inside the string *value* as Latex (requires \$ \$ or \$\$ \$\$ and similar latex tags).

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

### **Methods**

init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
	continues on next nage

continues on next page

Table 34 – continued from previous page

handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re-
	ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
<pre>hold_trait_notifications()</pre>	Context manager for bundling trait change notifica-
	tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a
	trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is con-
	structed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the
	widget.
<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the front-
	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
	end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only at-
	tributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

## **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been
	typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## \_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

## add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

## add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

## **classmethod class\_own\_trait\_events**(name: str) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

### **classmethod class\_trait\_names(\*\****metadata: Any*) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

#### classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

## description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated:: 8.0.0

Use tooltip attribute instead.

# focus()

Focus on the widget.

#### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

## **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

## placeholder

Placeholder text to display when nothing has been typed

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

## set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

#### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) \rightarrow dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.PrepareWidget.Layout

```
class AFL.automation.prepare.PrepareWidget.Layout(**kwargs: Any)
```

Bases: Widget

Layout specification

Defines a layout that can be expressed using CSS. Supports a subset of https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

When a property is also accessible via a shorthand property, we only expose the shorthand.

For example: - flex-grow, flex-shrink and flex-basis are bound to flex. - flex-wrap and flex-direction are bound to flex-flow. - margin-[top/bottom/left/right] values are bound to margin, etc.

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

Public constructor

# Methods

t t . White	D 111
init(**kwargs)	Public constructor
add_traits(**traits)	Dynamically add trait attributes to the Widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
	continues on poyt page

continues on next page

Table 35 – continued from previous page

trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

align_content	The align-content CSS attribute.
align_items	The align-items CSS attribute.
align_self	The align-self CSS attribute.
border	border property getter.
border_bottom	The border bottom CSS attribute.
border_left	The border left CSS attribute.
border_right	The border right CSS attribute.
border_top	The border top CSS attribute.
bottom	The bottom CSS attribute.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
display	The display CSS attribute.
flex	The flex CSS attribute.
flex_flow	The flex-flow CSS attribute.
grid_area	The grid-area CSS attribute.
grid_auto_columns	The grid-auto-columns CSS attribute.
grid_auto_flow	The grid-auto-flow CSS attribute.
grid_auto_rows	The grid-auto-rows CSS attribute.
grid_column	The grid-column CSS attribute.
grid_gap	The grid-gap CSS attribute.
grid_row	The grid-row CSS attribute.
grid_template_areas	The grid-template-areas CSS attribute.
grid_template_columns	The grid-template-columns CSS attribute.
grid_template_rows	The grid-template-rows CSS attribute.
height	The height CSS attribute.
justify_content	The justify-content CSS attribute.
justify_items	The justify-items CSS attribute.
keys	The traits which are synced.
left	The left CSS attribute.
log	A trait whose value must be an instance of a specified
	class.
margin	The margin CSS attribute.
max_height	The max-height CSS attribute.
max_width	The max-width CSS attribute.
min_height	The min-height CSS attribute.
min_width	The min-width CSS attribute.
model_id	Gets the model id of this widget.
object_fit	The object-fit CSS attribute.
object_position	The object-position CSS attribute.
order	The order CSS attribute.
order	The order CSS attribute.

continues on next page

Table 36 – continued from previous page

overflow	The overflow CSS attribute.
padding	The padding CSS attribute.
right	The right CSS attribute.
top	The top CSS attribute.
visibility	The visibility CSS attribute.
widget_types	
widgets	
width	The width CSS attribute.

# align\_content

The align-content CSS attribute.

## align\_items

The align-items CSS attribute.

# align\_self

The align-self CSS attribute.

## border\_top

The border top CSS attribute.

## border\_right

The border right CSS attribute.

# border\_bottom

The border bottom CSS attribute.

## border\_left

The border left CSS attribute.

## bottom

The bottom CSS attribute.

### display

The display CSS attribute.

# flex

The flex CSS attribute.

## flex\_flow

The flex-flow CSS attribute.

## height

The height CSS attribute.

## justify\_content

The justify-content CSS attribute.

# justify\_items

The justify-items CSS attribute.

# left

The left CSS attribute.

### margin

The margin CSS attribute.

# max\_height

The max-height CSS attribute.

### max\_width

The max-width CSS attribute.

## min\_height

The min-height CSS attribute.

## min\_width

The min-width CSS attribute.

### overflow

The overflow CSS attribute.

## order

The order CSS attribute.

## padding

The padding CSS attribute.

## right

The right CSS attribute.

## top

The top CSS attribute.

# visibility

The visibility CSS attribute.

## width

The width CSS attribute.

# object\_fit

The object-fit CSS attribute.

# object\_position

The object-position CSS attribute.

## grid\_auto\_columns

The grid-auto-columns CSS attribute.

## grid\_auto\_flow

The grid-auto-flow CSS attribute.

## grid\_auto\_rows

The grid-auto-rows CSS attribute.

## grid\_gap

The grid-gap CSS attribute.

# grid\_template\_rows

The grid-template-rows CSS attribute.

#### grid\_template\_columns

The grid-template-columns CSS attribute.

## grid\_template\_areas

The grid-template-areas CSS attribute.

### grid\_row

The grid-row CSS attribute.

#### grid\_column

The grid-column CSS attribute.

## grid\_area

The grid-area CSS attribute.

```
__init__(**kwargs)
```

Public constructor

### property border

border property getter. Return the common value of all side borders if they are identical. Otherwise return None.

```
__del__()
```

Object disposal

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## **classmethod class\_own\_trait\_events**(name: str) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

```
classmethod class_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

### Returns

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

#### get\_view\_spec()

# static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

#### on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- **name** (*list*, *str*, *None*) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

**unobserve**(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.PrepareWidget.PrepareWidget

#### **Methods**

```
SampleSeriesTool_reset_cb(event)

StockBuilder_reset_cb(event)

SweepBuilder_reset_cb(event)

__init__()

start()
```

```
__init__()
SweepBuilder_reset_cb(event)
StockBuilder_reset_cb(event)
SampleSeriesTool_reset_cb(event)
start()
```

# AFL.automation.prepare.PrepareWidget.PrepareWidget\_Model

class AFL.automation.prepare.PrepareWidget.PrepareWidget\_Model
 Bases: object
 \_\_init\_\_()

### **Methods**

\_\_init\_\_()

# AFL.automation.prepare.PrepareWidget.PrepareWidget\_View

\_\_init\_\_()

### **Methods**

```
__init__()
start(deck_builder_widget)
```

start(deck\_builder\_widget)

# AFL.automation.prepare.PrepareWidget.SampleSeriesWidget

### **Methods**

```
_init__(deck)
 apply_protocol_order()
 apply_protocol_order_cb(event)
 build_label(index)
 example_label_cb(event)
 make_all_labels()
 make_all_labels_cb(event)
 make_catch_protocol()
 make_mixing_wells()
 make_protocol()
 reset_uuid_cb(event)
 start()
 submit_cb(event)
 submit_mixing_wells_cb(event)
 sync_to_prepare_cb(event)
 update_mixing_well_preview_cb(event)
 update_protocol_order_preview_cb(event)
 update_sort_order_cb(event, direction)
__init__(deck)
apply_protocol_order_cb(event)
apply_protocol_order()
update_protocol_order_preview_cb(event)
make_protocol()
make_catch_protocol()
submit_cb(event)
```

```
build_label(index)
example_label_cb(event)
make_all_labels_cb(event)
make_all_labels()
sync_to_prepare_cb(event)
reset_uuid_cb(event)
update_sort_order_cb(event, direction)
make_mixing_wells()
update_mixing_well_preview_cb(event)
submit_mixing_wells_cb(event)
start()
```

## AFL.automation.prepare.PrepareWidget.StockBuilderWidget

### **Methods**

```
__init__(deck)

add_stocks_to_deck()

analyze_stocks_cb(event)

get_stock_objects()

get_stock_values()

load_cb(event)

make_stock_cb(event)

remove_stock_cb(event)

save_cb(event[, pkl])

set_units_cb(event, units)

start()

update_location_check_cb(event)
```

```
__init__(deck)
analyze_stocks_cb(event)
get_stock_objects()
add_stocks_to_deck()
get_stock_values()
save_cb(event, pkl=True)
load_cb(event)
update_location_check_cb(event)
make_stock_cb(event)
remove_stock_cb(event)
set_units_cb(event, units)
start()
```

# AFL.automation.prepare.PrepareWidget.SweepBuilderWidget

# **Methods**

```
__init__(deck)

calc_sweep_cb(click)

get_deck()

get_sweep_data()

plot_binary_cb(click)

plot_ternary_cb(click)

start()

update_component_row_cb(event)

validate_sweep_cb(click)
```

\_\_init\_\_(deck)

```
plot_binary_cb(click)
plot_ternary_cb(click)
calc_sweep_cb(click)
validate_sweep_cb(click)
get_sweep_data()
get_deck()
update_component_row_cb(event)
start()
```

# AFL.automation.prepare.PrepareWidget.Text

```
class AFL.automation.prepare.PrepareWidget.Text(**kwargs: Any)
    Bases: _String
    Single line textbox widget.
    __init__(*args, **kwargs)
    Public constructor
```

## **Methods**

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received
	continues on next page

Table 37 – continued from previous page

has_trait(name)	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
<pre>observe(handler[, names, type])</pre>	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_submit(callback[, remove])</pre>	(Un)Register a callback to handle text submission.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the frontend.
<pre>send_state([key])</pre>	Sends the widget state, or a piece of it, to the front- end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

comm	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use
	tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been
	typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

## disabled

Enable or disable user changes

## continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

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

Public constructor

## on\_submit(callback, remove=False)

(Un)Register a callback to handle text submission.

Triggered when the user clicks enter.

## **Parameters**

- callback (callable) Will be called with exactly one argument: the Widget instance
- remove (bool (optional)) Whether to unregister the callback

# \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# **classmethod class\_own\_traits**(\*\**metadata:* Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

## comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

### property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

#### focus()

Focus on the widget.

#### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

 $\textbf{key} \, (unicode \ or \ iterable \ (optional)) - A \ single \ property's \ name \ or \ iterable \ of \ property \ names \ to \ get.$ 

#### **Returns**

- **state** (dict of states)
- metadata (dict) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

### placeholder

Placeholder text to display when nothing has been typed

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

#### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

#### setup\_instance(\*\*kwargs: Any) $\rightarrow$ None

This is called **before** self.\_\_init\_\_ is called.

## tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.PrepareWidget.VBox

Displays multiple widgets vertically using the flexible box model.

### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '.' Applies a predefined style to the box. Defaults to ', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Vertical Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.VBox([title_widget, slider])
```

 $\verb|\__init__(children=(), **kwargs)|$ 

Public constructor

## **Methods**

add_class(class(className)       Adds a class to the top level element of the widget.         add_traits(**traits)       Dynamically add trait attributes to the Widget.         blur()       Blur the widget.         class_own_trait_events(name)       Get a dict of all event handlers defined on this class not a parent.         class_trait_names(**metadata)       Get a dict of all the traitlets defined on this class, no a parent.         class_trait_names(**metadata)       Get a list of all the names of this class' traits.         close()       Close method.         close()       Close method.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler],	init([children])	Public constructor
add_traits(**traits)         Dynamically add trait attributes to the Widget.           blur()         Blur the widget.           class_own_trait_events(name)         Get a dict of all event handlers defined on this class not a parent.           class_own_traits(**metadata)         Get a dict of all the traitlets defined on this class, no a parent.           class_trait_names(**metadata)         Get a list of all the names of this class' traits.           close()         Close method.           close_all()         Focus on the widget.           focus()         Focus on the widget.           get_manager_state([drop_defaults, widgets])         Gets the widget state, or a piece of it.           get_view_spec()         Returns the full state for a widget manager for embedding           handle_comm_opened(comm, msg)         Static method, called when a widget is constructed.           handle_control_comm_opened(comm, msg)         Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received           has_trait(name)         Returns True if the object has a trait with the specified name.           hold_sync()         Hold syncing any state until the outermost contex manager exits           hold_trait_notifications()         Context manager for bundling trait change notifications and cross validation.           notify_change(change)         Setup a handler to be called when a trait changes.           on_		
blur()  class_own_trait_events(name)  class_own_traits(**metadata)  class_trait_names(**metadata)  class_traits(**metadata)  class_traits(**metadata)  close()  close_all()  focus()  get_manager_state([drop_defaults, widgets])  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  hold_sync()  hold_trait_notifications()  notify_change(change)  observe(handler, names, type])  on_trait_change([handler, name, remove])  open()  open()  open a comm to the frontend if one isn't already open remove_class(className)  et a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the traitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class, not a parent.  Get a dict of all the raitlets defined on this class.  Close method.  Close method.  Close method.  Close method.  Close method.  Clast weidget state, or a piece of it.   Static method, called when a widget is constructed.  Class method, called when a strait thange notification and cross validation.  Class method, called when a property has charged the parent of the decived on the parent of the parent of the parent of the decived of the parent of the decivery of the parent of the par		
class_owm_trait_events(name)       Get a dict of all event handlers defined on this class not a parent.         class_owm_traits(**metadata)       Get a dict of all the traitlets defined on this class, no a parent.         class_trait_names(**metadata)       Get a list of all the names of this class' traits.         close()       Close all (or all the traits of this class.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Focus on the widget state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Class method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       Cln/Register a custom msg receive callback.         on_widget_constructed(callback)       Registers a callback t	· · · · · · · · · · · · · · · · · · ·	•
not a parent.  Get a dict of all the traitlets defined on this class, no a parent.  Class_trait_names(**metadata)  Class_trait_s(**metadata)  Close()  Close adict of all the traitlets of this class' traits.  Close()  Close method.  Close method.  Close method.  Focus on the widget.  Returns the full state for a widget manager for embedding  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  Returns True if the object has a trait with the specified name.  hold_sync()  Hold syncing any state until the outermost contex manager exits  hold_trait_notifications()  notify_change(change)  observe(handler[, names, type])  on_msg(callback[, remove])  on_trait_change([handler, name, remove])  open()  open()  Open a comm to the frontend if one isn't already open remove_class(className)  Returns True if the object has a trait when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_class(className)  Returns True if the object has a trait with the specified name.  Called when a property has changed.  Setup a handler to be called when a trait changes.  (Un)Register a custom msg receive callback.  Registers a callback to be called when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_class(className)  Removes a class from the top level element of the	· · · · · · · · · · · · · · · · · · ·	ē
a parent.  Class_trait_names(**metadata)		not a parent.
class_traits(**metadata)       Get a dict of all the traits of this class.         close()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	class_own_traits(**metadata)	
close()       Close method.         close_all()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
Focus on the widget.   Gets manager_state([drop_defaults, widgets])   Returns the full state for a widget manager for embedding   Gets the widget state, or a piece of it.	class_traits(**metadata)	Get a dict of all the traits of this class.
focus()  get_manager_state([drop_defaults, widgets])  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  has_trait(name)  hold_sync()  hold_trait_notifications()  notify_change(change)  observe(handler[, names, type])  on_msg(callback[, remove])  on_widget_constructed(callback)  on_widget_constructed(callback)  open()  remove_class(className)  Focus on the widget.  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget manager for embedding  Returns the full state for a widget is constructed.  Class method, called when a widget is constructed.  Class method, called when a trait with the specified name.  Hold_sync()  Hold syncing any state until the outermost contex manager exits  Context manager for bundling trait change notifications and cross validation.  Called when a property has changed.  Setup a handler to be called when a trait changes.  On_msg(callback[, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_class(className)	close()	Close method.
get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)       Removes a class from the top level element of the	close_all()	
ding  get_state([key, drop_defaults])  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  has_trait(name)  hold_sync()  hold_trait_notifications()  conserve(handler[, names, type])  observe(handler[, names, type])  on_msg(callback[, remove])  on_widget_constructed(callback)  on_widget_constructed(callback)  remove_class(className)  ding  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Gets the widget state, or a piece of it.  Glass method, called when a widget is constructed.  Class method, called when a widget is constructed.  Open a comm to the frontend if one isn't already open remove_clas	V .	<del>_</del>
get_view_spec()         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)	<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost contex manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)       Removes a class from the top level element of the	<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open Remove_class(className)	<pre>get_view_spec()</pre>	
on the "jupyter.widget.control" comm channel is received  has_trait(name)  Returns True if the object has a trait with the specified name.  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Called when a property has changed.  observe(handler[, names, type])  Setup a handler to be called when a trait changes.  on_msg(callback[, remove])  (Un)Register a custom msg receive callback.  on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  on_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  open()  Open a comm to the frontend if one isn't already open remove_class(className)  Removes a class from the top level element of the	<pre>handle_comm_opened(comm, msg)</pre>	
name.  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open Remove_class(className)  Removes a class from the top level element of the	handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  Om_msg(callback[, remove])  Called when a property has changed.  Setup a handler to be called when a trait changes.  (Un)Register a custom msg receive callback.  Om_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  Om_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open remove_class(className)  Removes a class from the top level element of the	has_trait(name)	Returns True if the object has a trait with the specified name.
hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  On_widget_constructed(callback)  Open()  Open a comm to the frontend if one isn't already open Remove_class(className)  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Called when a property has changed.  Other trait change notifications and cross validation.  Open a common to the frontend if one isn't already open notifications and cross validation.	hold_sync()	Hold syncing any state until the outermost context manager exits
observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the	hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the	notify_change(change)	Called when a property has changed.
on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		
on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		
on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open remove_class(className)         Removes a class from the top level element of the		DEPRECATED: Setup a handler to be called when a
remove_class(className) Removes a class from the top level element of the	on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
	open()	Open a comm to the frontend if one isn't already open.
widget.	remove_class(className)	Removes a class from the top level element of the widget.

continues on next page

Table 38 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

### **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
   Object disposal
__init__(children=(), **kwargs)
   Public constructor

add_class(className)
   Adds a class to the top level element of the widget.
   Doesn't add the class if it already exists.

add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

#### box\_style

Use a predefined styling for the box.

#### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

### classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

### **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

#### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

#### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

## tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

## **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

## **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
    widgets = {}

class AFL.automation.prepare.PrepareWidget.PrepareWidget
    __init__()
    SweepBuilder_reset_cb(event)
    StockBuilder_reset_cb(event)
    SampleSeriesTool_reset_cb(event)
    start()

class AFL.automation.prepare.PrepareWidget.PrepareWidget_Model

class AFL.automation.prepare.PrepareWidget.PrepareWidget_View
    start(deck_builder_widget)
```

# AFL.automation.prepare.SampleSeriesWidget

#### **Functions**

sqrt(x,/)

Return the square root of x.

## AFL.automation.prepare.SampleSeriesWidget.sqrt

```
AFL.automation.prepare.SampleSeriesWidget.\mathbf{sqrt}(x,/)
Return the square root of x.
```

### **Classes**

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean <i>value</i> in the form of a checkbox.
<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
FloatText(**kwargs)	Displays a float value within a textbox.
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible
	box model.
IntText(**kwargs)	Textbox widget that represents an integer.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
SampleSeriesWidget(deck)	
SampleSeriesWidget_Model(deck)	
SampleSelleSwluget_Model(deck)	
SampleSeriesWidget_View()	
- v	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible
	box model.

## AFL.automation.prepare.SampleSeriesWidget.Button

class AFL.automation.prepare.SampleSeriesWidget.Button(\*\*kwargs: Any)

Bases: DOMWidget, CoreWidget

Button widget.

This widget has an  $on\_click$  method that allows you to listen for the user clicking on the button. The click event itself is stateless.

#### **Parameters**

- **description** (*str*) description displayed on the button
- icon (str) font-awesome icon names, without the 'fa-' prefix
- **disabled** (bool) whether user interaction is enabled

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

Public constructor

## **Methods**

init(**kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not
	a parent.

continues on next page

Table 39 – continued from previous page

	ed from previous page
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
click()	Programmatically trigger a click event.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
<pre>handle_control_comm_opened(comm, msg)</pre>	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
has_trait(name)	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
<pre>hold_trait_notifications()</pre>	Context manager for bundling trait change notifica-
	tions and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
<pre>on_click(callback[, remove])</pre>	Register a callback to execute when the button is
	clicked.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a
	trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is con-
	structed.
open()	Open a comm to the frontend if one isn't already open.
<pre>remove_class(className)</pre>	Removes a class from the top level element of the
	widget.
send(content[, buffers])	Sends a custom msg to the widget model in the front-
	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
	end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only at-
	tributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.

continues on next page

Table 39 - continued from previous page

1 7.7(5 3)	
unobserve_all([name])	Remove trait change handlers of any type for the spec-
unobserve_arr([name])	Remove trait change handlers of any type for the spec-
	:C - 1
	ified name.
	11100 11011101

### **Attributes**

button_style	Use a predefined styling for the button.
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Button label.
disabled	Enable or disable user changes.
icon	Font-awesome icon names, without the 'fa-' prefix.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

# description

Button label.

### disabled

Enable or disable user changes.

## icon

Font-awesome icon names, without the 'fa-' prefix.

## button\_style

Use a predefined styling for the button.

### style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# \_\_init\_\_(\*\*kwargs)

Public constructor

# on\_click(callback, remove=False)

Register a callback to execute when the button is clicked.

The callback will be called with one argument, the clicked button widget instance.

## **Parameters**

**remove** (bool (optional)) – Set to true to remove the callback from the list of callbacks.

### click()

Programmatically trigger a click event.

This will call the callbacks registered to the clicked button widget instance.

## \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

## add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

### blur()

Blur the widget.

## classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the *trait\_names()* method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

#### classmethod close\_all()

#### comm

A trait which allows any value.

#### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

### get\_view\_spec()

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default:* '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

#### set\_state(sync data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

### tabbable

Is widget tabbable?

## tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

 $trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any$ 

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getatr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.Checkbox

Displays a boolean value in the form of a checkbox.

### **Parameters**

- value ({True, False}) value of the checkbox: True-checked, False-unchecked
- **description** (*str*) description displayed next to the checkbox
- **indent** ({*True*, *False*}) indent the control to align with other controls with a description. The style description width attribute controls this width for consistence with other controls.

```
__init__(value=None, **kwargs)
```

Public constructor

575

# Methods

init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be
	passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
<pre>handle_control_comm_opened(comm, msg)</pre>	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica-
	tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is con-
-	structed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the
,	widget.
send(content[, buffers])	Sends a custom msg to the widget model in the front-
	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
	end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
	continues on next page
	continues on next page

Table 40 – continued from previous page

trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use
	tooltip attribute instead.
disabled	Enable or disable user changes.
indent	Indent the control to align with other controls with a
	description.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Bool value
widget_types	
widgets	
Š	

# indent

Indent the control to align with other controls with a description.

# style

Styling customizations

\_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the *trait\_names()* method, but is unbound.

# classmethod class\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

# description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

### disabled

Enable or disable user changes.

### focus()

Focus on the widget.

## get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

## Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

### **Returns**

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

# $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

# layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (*list of binary buffers*) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

## tabbable

Is widget tabbable?

# tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

### value

Bool value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
```

## **Methods**

```
_init__([ip, port, username, interactive])
clear_history()
clear_queue()
debug(state)
deposit_obj(obj[, uid])
                                                  Deposit an object in the dropbox obj: object, the ob-
                                                  ject to deposit id: str, the uuid to deposit the object
                                                  under if not specified, a new uuid will be generated
driver_status()
enqueue([interactive])
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
get_unqueued_commands([inherit_commands])
halt()
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
```

continues on next page

Table 41 – continued from previous page

```
remove_item(uuid)
 reset_queue_daemon()
                                                  Retrieve an object from the dropbox id: str, the uuid
 retrieve_obj(uid[, delete])
                                                  of the object to retrieve delete: bool, if True, delete
                                                  the object after retrieving
 server_cmd(cmd, **kwargs)
 set_config([interactive])
 set_driver_object(**kw)
 set_object([serialize])
 unqueued_base(**kwargs)
 wait([target_uuid, interval, for_history, ...])
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
```

```
reset_queue_daemon()
pause(state)
clear_queue()
clear_history()
debug(state)
halt()
queue_state()
remove_item(uuid)
move_item(uuid, pos)
set_driver_object(**kw)
get_driver_object(name)
deposit_obj(obj, uid=None)
     Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
     under if not specified, a new uuid will be generated
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
     the object after retrieving
set_object(serialize=True, **kw)
get_object(name, serialize=True)
```

## AFL.automation.prepare.SampleSeriesWidget.FloatText

Displays a float value within a textbox. For a textbox in which the value must be within a specific range, use BoundedFloatText.

### **Parameters**

- value (float) value displayed
- **step** (*float*) step of the increment (if None, any step is allowed)
- **description** (*str*) description displayed next to the text box

```
__init__(value=None, **kwargs)
Public constructor
```

# Methods

, , , (F. 1. 1)	D 11:
init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	•
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
<pre>observe(handler[, names, type])</pre>	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
	continues on next page

continues on next page

Table 42 – continued from previous page

trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

comm	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use
	tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
step	Minimum step to increment the value
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Float value
widget_types	
widgets	

# disabled

Enable or disable user changes

# continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

# step

Minimum step to increment the value

\_\_del\_\_()
Object disposal
\_\_init\_\_(value=None, \*\*kwargs)
Public constructor

### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

## comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

## property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

### focus()

Focus on the widget.

### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

### **Returns**

- **state** (dict of states)
- metadata (dict) metadata for each field: {key: metadata}

```
get_view_spec()
```

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

### **send**(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

## **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

### Parameters

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

### style

Styling customizations

### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

### value

Float value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.HBox

```
class AFL.automation.prepare.SampleSeriesWidget.HBox(**kwargs: Any)
    Bases: Box
```

Displays multiple widgets horizontally using the flexible box model.

### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '.' Applies a predefined style to the box. Defaults to ', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Horizontal Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.HBox([title_widget, slider])
```

 $\verb"\__init__(children=(), **kwargs")$ 

Public constructor

# **Methods**

init([children])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
<pre>on_widget_constructed(callback)</pre>	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.

continues on next page

Table 43 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
   Object disposal
__init__(children=(), **kwargs)
   Public constructor

add_class(className)
   Adds a class to the top level element of the widget.
   Doesn't add the class if it already exists.

add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

### blur()

Blur the widget.

### box\_style

Use a predefined styling for the box.

### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

### focus()

Focus on the widget.

# static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

# log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

### tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

## **Parameters**

**name** (*str* (*default: None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# Notes

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

## **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.SampleSeriesWidget.IntText

```
class AFL.automation.prepare.SampleSeriesWidget.IntText(**kwargs: Any)
    Bases: _Int
    Textbox widget that represents an integer.
    __init__(value=None, **kwargs)
    Parameters
        value(integer) - The initial value.
```

### **Methods**

init([value])	
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	

continues on next page

Table 44 – continued from previous page

handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open.         remove_class(className)       Removes a class from the top level element of the widget.         send_state([key])       Sends a custom msg to the widget model in the frontend, if it exists.         set_state(sync_data)       Called when a state is received from the frontend, if it exists.         set_rait_(name, value)       Forcibly sets trait attribute, including read-only attributes.         setup_instance(**kwargs)       This is called before selfinit is called.		<u></u>
on the "jupyter.widget.control" comm channel is received  Returns True if the object has a trait with the specified name.  hold_sync() Hold syncing any state until the outermost context manager exits  hold_trait_notifications() Context manager for bundling trait change notifications and cross validation.  notify_change(change) Observe(handler[, names, type]) On_msg(callback[, remove]) ClupRegister a custom msg receive callback.  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback) Registers a callback to be called when a widget is constructed.  Open() Open a comm to the frontend if one isn't already open.  remove_class(className) Removes a class from the top level element of the widget.  send(content[, buffers]) Sends a custom msg to the widget model in the frontend.  send_state([key]) Sends the widget state, or a piece of it, to the frontend, if it exists.  called when a state is received from the frontend, if it exists.  Called when a state is received from the frontend, if it exists.  called when a state is received from the frontend.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs) This is called before selfinit is called.  Return a trait's default value or a dictionary of them trait_events([name])  trait_mames(**metadata) Returns True if the specified trait has a value.  Get a dict of all the event handlers of this class.  trait_names(**metadata) Get a dict of all the names of this class' traits.  trait_values(**metadata) Get a dict of all the names of this class.  trait_values(**metadata) Get a dict of all the traits of this class.  trait_values(**metadata) Get a dict of all the traits of this class.  trait_values(**metadata) Get a dict of all the traits of this class.  trait_values(**metadata) Get a dict of all the traits of this class.  trait_values(**metadata) Get a dict of all the traits of this class.  trait_values(**metadata) Get a dict of all the	handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
name.  hold_sync()  hold_sync()  hold_syncing any state until the outermost context manager exits  Context manager for bundling trait change notifications and cross validation.  Called when a property has changed.  observe(handler[, names, type])  observe(handler[, names, type])  observe(handler[, names, type])  observe(handler[, name, remove])  observe(handler[, name, type])  called when a property has changed.  Called when a property has changed.  Called when a property has changed.  Ocalled when a property has changed.  Ocalled when a property has changed.  Observe(handler[, name, stype])  Remove a class from the ob called when a widget is constructed.  Open a comm to the frontend if one isn't already open.  Removes a class from the top level element of the widget.  Sends a custom msg to the widget model in the frontend.  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Called when a state is received from the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Sends t	handle_control_comm_opened(comm, msg)	on the "jupyter.widget.control" comm channel is re-
manager exits Context manager for bundling trait change notifications and cross validation.  notify_change(change) Called when a property has changed.  Setup a handler to be called when a trait changes.  on_msg(callback[, remove]) Open(longuidet_constructed(callback)  nemove_class(className)  send_state([key])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends a custom msg to the widget model in the frontend.  set_trait(name, value)  set_p_instance(**kwargs)  trait_defaults(*name, **metadata)  trait_mames(**metadata)  trait_values(**metadata)  trait_semove_tandler[, names, type])  manager exits Context manager for bundling trait change notifications and cross validation.  Called when a property has changed.  Setup a handler to be called when a trait changes.  (Un)Register a custom msg receive callback.  DEPRECATED: Setup a handler to be called when a widget is constructed.  Open ()  Open a comm to the frontend if one isn't already open.  Removes a class from the top level element of the widget.  Sends a custom msg to the widget model in the frontend.  Send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  Called when a state is received from the frontend.  Set_state(sync_data)  Set_state(sync_data)  Set_state(sync_data)  Set a state is received from the frontend.  Set_rait_defaults(*name, value)  Forcibly sets trait attribute, including read-only attributes.  Setup_instance(**kwargs)  This is called before selfinit is called.  Return a trait's default value or a dictionary of them  Get a dict of all the event handlers of this class.  Get a list of all the names of this class' traits.  A dict of rait names and their values.  Get a list of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handlers of any type for the spec-	has_trait(name)	
tions and cross validation.  Called when a property has changed.  Observe(handler[, names, type])  Setup a handler to be called when a trait changes.  On_msg(callback[, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  On_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  Open ()  Open a comm to the frontend if one isn't already open.  Removes a class from the top level element of the widget.  Send(content[, buffers])  Sends a custom msg to the widget model in the frontend, if it exists.  Set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  Setup_instance(**kwargs)  This is called before selfinit is called.  Trait_defaults(*names, **metadata)  Trait_has_value(name)  Return a trait's default value or a dictionary of them trait_events([name])  Get a dict of all the event handlers of this class.  Trait_names(**metadata)  Get a list of all the names of this class' traits.  Trait_values(**metadata)  Get a list of all the raits of this class.  Unobserve(handler[, names, type])  Remove trait change handlers  Remove trait change handlers of any type for the spec-	hold_sync()	
observe(handler[, names, type])         Setup a handler to be called when a trait changes.           on_msg(callback[, remove])         (Un)Register a custom msg receive callback.           on_trait_change([handler, name, remove])         DEPRECATED: Setup a handler to be called when a trait changes.           on_widget_constructed(callback)         Registers a callback to be called when a widget is constructed.           open()         Open a comm to the frontend if one isn't already open.           remove_class(className)         Removes a class from the top level element of the widget.           send_content[, buffers])         Sends a custom msg to the widget model in the frontend.           send_state([key])         Sends the widget state, or a piece of it, to the frontend, if it exists.           set_state(sync_data)         Called when a state is received from the frontend.           set_rait(name, value)         Forcibly sets trait attribute, including read-only attributes.           setup_instance(**kwargs)         This is called before selfinit is called.           trait_defaults(*names, **metadata)         Return a trait's default value or a dictionary of them           trait_events([name])         Get a dict of all the event handlers of this class.           trait_names(**metadata(traitname, key[, default])         Get metadata values for trait by key.           trait_values(**metadata)         Get a list of all the names of this class' traits.           tr	hold_trait_notifications()	
on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the frontend.set_state(sync_data)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*name, **metadata)Return a trait's default value or a dictionary of them trait_events([name])trait_has_value(name)Returns True if the specified trait has a value.trait_matedata(traitname, key[, default])Get a dict of all the event handlers of this class.trait_values(**metadata)Get a list of all the names of this class' traits.traits(**metadata)Get a list of all the traits of this class.unobserve(handler[, names, type])Remove a trait change handler.unobserve_all([name])Remove trait change handlers of any type for the spec-	<pre>notify_change(change)</pre>	Called when a property has changed.
on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit_ is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.trait_metadata(traitname, key[, default])Get metadata values for trait by key.trait_names(**metadata)Get a list of all the mames of this class' traits.traits(**metadata)Get a dict of all the traits of this class.unobserve(handler[, names, type])Remove a trait change handler.unobserve_all([name])Remove trait change handlers of any type for the spec-	observe(handler[, names, type])	Setup a handler to be called when a trait changes.
trait changes.  on_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  open()  Open a comm to the frontend if one isn't already open.  Removes a class from the top level element of the widget.  Sends a custom msg to the widget model in the frontend.  Send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  Setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  trait_names(**metadata)  Return a trait's default value or a dictionary of them trait_events([name])  Get a dict of all the event handlers of this class.  trait_names(**metadata)  Get metadata values for trait by key.  trait_names(**metadata)  Get a dict of all the names of this class' traits.  trait_values(**metadata)  Get a dict of all the traits of this class' traits.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handlers of any type for the spec-	<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
open() open() Open a comm to the frontend if one isn't already open. remove_class(className) Removes a class from the top level element of the widget. send(content[, buffers]) Sends a custom msg to the widget model in the frontend. send_state([key]) Sends the widget state, or a piece of it, to the frontend, if it exists. set_state(sync_data) Set_trait(name, value) Forcibly sets trait attribute, including read-only attributes. setup_instance(**kwargs) This is called before selfinit is called. trait_defaults(*names, **metadata) Return a trait's default value or a dictionary of them trait_events([name]) Get a dict of all the event handlers of this class. trait_metadata(traitname, key[, default]) Get metadata values for trait by key. trait_names(**metadata) Get a list of all the names of this class' traits. trait_values(**metadata) Get a dict of all the traits of this class. traits(**metadata) Get a dict of all the traits of this class. unobserve(handler[, names, type]) Remove a trait change handler. unobserve_all([name]) Remove trait change handlers of any type for the spec-	<pre>on_trait_change([handler, name, remove])</pre>	•
remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.trait_metadata(traitname, key[, default])Get metadata values for trait by key.trait_names(**metadata)Get a list of all the names of this class' traits.trait_values(**metadata)A dict of trait names and their values.traits(**metadata)Get a dict of all the traits of this class.unobserve(handler[, names, type])Remove a trait change handler.unobserve_all([name])Remove trait change handlers of any type for the spec-	<pre>on_widget_constructed(callback)</pre>	
widget.  send(content[, buffers])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  called when a state is received from the frontend.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  trait_events([name])  Get a dict of all the event handlers of this class.  trait_metadata(traitname, key[, default])  trait_names(**metadata)  Get metadata values for trait by key.  trait_names(**metadata)  Get a list of all the names of this class' traits.  trait_values(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handlers  Remove trait change handlers of any type for the spec-	open()	Open a comm to the frontend if one isn't already open.
send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  Set_state(sync_data)  Set_trait(name, value)  Set_up_instance(**kwargs)  trait_defaults(*names, **metadata)  trait_events([name])  This is called before selfinit is called.  Return a trait's default value or a dictionary of them trait_events([name])  Get a dict of all the event handlers of this class.  trait_metadata(traitname, key[, default])  trait_metadata(traitname, key[, default])  Get a list of all the names of this class' traits.  trait_values(**metadata)  Get a list of all the traits of this class' traits.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	remove_class(className)	
end, if it exists.  Set_state(sync_data)  Set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  Setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.  trait_has_value(name)  trait_metadata(traitname, key[, default])  Get metadata values for trait by key.  trait_names(**metadata)  Get a list of all the names of this class' traits.  trait_values(**metadata)  A dict of trait names and their values.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  unobserve_all([name])  Remove trait change handlers of any type for the spec-	send(content[, buffers])	<u> </u>
Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  trait_defaults(*names, **metadata)  trait_events([name])  trait_has_value(name)  trait_metadata(traitname, key[, default])  trait_names(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_values(**metadata)  trait_s(**metadata)  Get a dict of all the names of this class' traits.  A dict of trait names and their values.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	send_state([key])	
tributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.  trait_has_value(name)  trait_metadata(traitname, key[, default])  Get metadata values for trait by key.  trait_names(**metadata)  Get a list of all the names of this class' traits.  trait_values(**metadata)  A dict of trait names and their values.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  unobserve_all([name])  Remove trait change handlers of any type for the spec-	<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.  trait_has_value(name)  trait_metadata(traitname, key[, default])  Get metadata values for trait by key.  trait_names(**metadata)  Get a list of all the names of this class' traits.  trait_values(**metadata)  A dict of trait names and their values.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	set_trait(name, value)	·
trait_events([name])  trait_has_value(name)  trait_has_value(name)  trait_metadata(traitname, key[, default])  trait_names(**metadata)  trait_values(**metadata)  trait_values(**metadata)  traits(**metadata)  Get a dict of all the names of this class' traits.  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_has_value(name)  trait_metadata(traitname, key[, default])  trait_metadata(traitname, key[, default])  trait_names(**metadata)  trait_values(**metadata)  traits(**metadata)  traits(**metadata)  traits(**metadata)  unobserve(handler[, names, type])  unobserve_all([name])  Returns True if the specified trait has a value.  Get metadata values for trait by key.  Get a list of all the names of this class' traits.  A dict of trait names and their values.  Get a dict of all the traits of this class.  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
trait_metadata(traitname, key[, default])  trait_names(**metadata)  trait_values(**metadata)  trait_values(**metadata)  traits(**metadata)  traits(**metadata)  unobserve(handler[, names, type])  unobserve_all([name])  Get metadata values for trait by key.  Get a list of all the names of this class' traits.  A dict of trait names and their values.  Get a dict of all the traits of this class.  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	<pre>trait_events([name])</pre>	Get a dict of all the event handlers of this class.
trait_names(**metadata)  trait_values(**metadata)  trait_values(**metadata)  traits(**metadata)  traits(**metadata)  unobserve(handler[, names, type])  unobserve_all([name])  Get a list of all the names of this class' traits.  A dict of trait names and their values.  Get a dict of all the traits of this class.  Remove a trait change handler.  Remove trait change handlers of any type for the spec-	trait_has_value(name)	Returns True if the specified trait has a value.
trait_values(**metadata)  traits(**metadata)  Get a dict of all the traits of this class.  unobserve(handler[, names, type])  unobserve_all([name])  Remove trait change handlers of any type for the spec-	<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
traits(**metadata)  unobserve(handler[, names, type])  unobserve_all([name])  Get a dict of all the traits of this class.  Remove a trait change handler.  Remove trait change handlers of any type for the spec-		
unobserve(handler[, names, type])Remove a trait change handler.unobserve_all([name])Remove trait change handlers of any type for the spec-		
unobserve_all([name]) Remove trait change handlers of any type for the spec-	· · · · · · · · · · · · · · · · · · ·	Get a dict of all the traits of this class.
		•
	unobserve_all([name])	

# **Attributes**

COMM	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use
	tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
step	Minimum step to increment the value
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Int value
widget_types	
widgets	

# disabled

Enable or disable user changes

# continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

# step

Minimum step to increment the value

\_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

# **Parameters**

**value** (*integer*) – The initial value.

## add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

### classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow list[str]$

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

## description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

## focus()

Focus on the widget.

### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

## Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | str
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default:* '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

# open()

Open a comm to the frontend if one isn't already open.

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

### style

Styling customizations

# tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

# trait\_defaults(\*names: str, \*\*metadata: Any) → dict[str, Any] | Sentinel

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Int value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.Label

```
class AFL.automation.prepare.SampleSeriesWidget.Label(**kwargs: Any)
    Bases: _String
    Label widget.

It also renders math inside the string value as Latex (requires $ $ or $$ $$ and similar latex tags).
    __init__(value=None, **kwargs)
    Public constructor
```

# Methods

, , , (F. 1. 1)	D 11:
init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	•
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
<pre>observe(handler[, names, type])</pre>	Setup a handler to be called when a trait changes.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
	continues on next page

continues on next page

Table 45 – continued from previous page

trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

\_\_del\_\_()
 Object disposal
\_\_init\_\_(value=None, \*\*kwargs)
 Public constructor
add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

#### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

### blur()

Blur the widget.

## classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the *trait\_names()* method, but is unbound.

## classmethod class\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

## description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

### focus()

Focus on the widget.

## get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode or iterable (optional*)) – A single property's name or iterable of property names to get.

#### Returns

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

```
static handle_comm_opened(comm, msg)
```

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

# hold\_sync()

Hold syncing any state until the outermost context manager exits

## $hold\_trait\_notifications() \rightarrow Any$

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

## placeholder

Placeholder text to display when nothing has been typed

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

# send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

## trait\_defaults(\*names: str, \*\*metadata: Any) → dict[str, Any] | Sentinel

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# **Return type**

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.Layout

```
class AFL.automation.prepare.SampleSeriesWidget.Layout(**kwargs: Any)
```

Bases: Widget

Layout specification

Defines a layout that can be expressed using CSS. Supports a subset of https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

When a property is also accessible via a shorthand property, we only expose the shorthand.

For example: - flex-grow, flex-shrink and flex-basis are bound to flex. - flex-wrap and flex-direction are bound to flex-flow. - margin-[top/bottom/left/right] values are bound to margin, etc.

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

Public constructor

# Methods

I I A Calculat	D.11
init(**kwargs)	Public constructor
add_traits(**traits)	Dynamically add trait attributes to the Widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
<pre>has_trait(name)</pre>	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
	continues on poyt page

continues on next page

Table 46 – continued from previous page

trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

align_content	The align-content CSS attribute.
align_items	The align-items CSS attribute.
align_self	The align-self CSS attribute.
border	border property getter.
border_bottom	The border bottom CSS attribute.
border_left	The border left CSS attribute.
border_right	The border right CSS attribute.
border_top	The border top CSS attribute.
bottom	The bottom CSS attribute.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
display	The display CSS attribute.
flex	The flex CSS attribute.
flex_flow	The flex-flow CSS attribute.
grid_area	The grid-area CSS attribute.
grid_auto_columns	The grid-auto-columns CSS attribute.
grid_auto_flow	The grid-auto-flow CSS attribute.
grid_auto_rows	The grid-auto-rows CSS attribute.
grid_column	The grid-column CSS attribute.
grid_gap	The grid-gap CSS attribute.
grid_row	The grid-row CSS attribute.
grid_template_areas	The grid-template-areas CSS attribute.
grid_template_columns	The grid-template-columns CSS attribute.
grid_template_rows	The grid-template-rows CSS attribute.
height	The height CSS attribute.
justify_content	The justify-content CSS attribute.
justify_items	The justify-items CSS attribute.
keys	The traits which are synced.
left	The left CSS attribute.
log	A trait whose value must be an instance of a specified
9	class.
margin	The margin CSS attribute.
max_height	The max-height CSS attribute.
max_width	The max-width CSS attribute.
min_height	The min-height CSS attribute.
min_width	The min-width CSS attribute.
model_id	Gets the model id of this widget.
object_fit	The object-fit CSS attribute.
object_position	The object-position CSS attribute.
order	The order CSS attribute.
	continues on port page

continues on next page

Table 47 – continued from previous page

overflow	The overflow CSS attribute.
padding	The padding CSS attribute.
right	The right CSS attribute.
top	The top CSS attribute.
visibility	The visibility CSS attribute.
widget_types	
widgets	
width	The width CSS attribute.

# align\_content

The align-content CSS attribute.

## align\_items

The align-items CSS attribute.

# align\_self

The align-self CSS attribute.

## border\_top

The border top CSS attribute.

## border\_right

The border right CSS attribute.

## border\_bottom

The border bottom CSS attribute.

## border\_left

The border left CSS attribute.

## bottom

The bottom CSS attribute.

## display

The display CSS attribute.

## flex

The flex CSS attribute.

## flex\_flow

The flex-flow CSS attribute.

## height

The height CSS attribute.

## justify\_content

The justify-content CSS attribute.

# justify\_items

The justify-items CSS attribute.

## left

The left CSS attribute.

## margin

The margin CSS attribute.

## max\_height

The max-height CSS attribute.

## max\_width

The max-width CSS attribute.

## min\_height

The min-height CSS attribute.

## min\_width

The min-width CSS attribute.

## overflow

The overflow CSS attribute.

## order

The order CSS attribute.

## padding

The padding CSS attribute.

## right

The right CSS attribute.

#### top

The top CSS attribute.

# visibility

The visibility CSS attribute.

## width

The width CSS attribute.

# object\_fit

The object-fit CSS attribute.

## object\_position

The object-position CSS attribute.

## grid\_auto\_columns

The grid-auto-columns CSS attribute.

## grid\_auto\_flow

The grid-auto-flow CSS attribute.

## grid\_auto\_rows

The grid-auto-rows CSS attribute.

## grid\_gap

The grid-gap CSS attribute.

# grid\_template\_rows

The grid-template-rows CSS attribute.

#### grid\_template\_columns

The grid-template-columns CSS attribute.

## grid\_template\_areas

The grid-template-areas CSS attribute.

### grid\_row

The grid-row CSS attribute.

### grid\_column

The grid-column CSS attribute.

### grid\_area

The grid-area CSS attribute.

```
__init__(**kwargs)
```

Public constructor

### property border

border property getter. Return the common value of all side borders if they are identical. Otherwise return None.

```
__del__()
```

Object disposal

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## **classmethod class\_own\_trait\_events**(name: str) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

```
classmethod class_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

### Returns

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

# static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

### on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- **name** (*list*, *str*, *None*) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

## **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) \rightarrow dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## **Return type**

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget

## **Methods**

```
_init__(deck)
 apply_protocol_order()
 apply_protocol_order_cb(event)
 build_label(index)
 example_label_cb(event)
 make_all_labels()
 make_all_labels_cb(event)
 make_catch_protocol()
 make_mixing_wells()
 make_protocol()
 reset_uuid_cb(event)
 start()
 submit_cb(event)
 submit_mixing_wells_cb(event)
 sync_to_prepare_cb(event)
 update_mixing_well_preview_cb(event)
 update_protocol_order_preview_cb(event)
 update_sort_order_cb(event, direction)
__init__(deck)
apply_protocol_order_cb(event)
apply_protocol_order()
update_protocol_order_preview_cb(event)
make_protocol()
make_catch_protocol()
submit_cb(event)
```

```
build_label(index)
    example_label_cb(event)
    make_all_labels_cb(event)
    make_all_labels()
    sync_to_prepare_cb(event)
    reset_uuid_cb(event)
    update_sort_order_cb(event, direction)
    make_mixing_wells()
    update_mixing_well_preview_cb(event)
    submit_mixing_wells_cb(event)
    start()
AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Model
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Model(deck)
    Bases: object
    __init__(deck)
    Methods
      __init__(deck)
      adjust_protocol_order(mix_order)
    __init__(deck)
    adjust_protocol_order(mix_order)
AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_View
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_View
    Bases: object
    __init__()
```

## **Methods**

```
__init__()

make_component_grid(components, nsamples)

make_mixing_wells()

make_pipette_params(name)

start(components, nsamples, stock_names)
```

```
__init__()
make_component_grid(components, nsamples)
make_pipette_params(name)
make_mixing_wells()
start(components, nsamples, stock_names)
```

# AFL.automation.prepare.SampleSeriesWidget.Text

## **Methods**

630

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not
	a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.

continues on next page

Table 48 – continued from previous page

ed from previous page
Return the value for this widget which should be passed to interactive functions.
Returns the full state for a widget manager for embedding
Gets the widget state, or a piece of it.
•
Static method, called when a widget is constructed.
Class method, called when the comm-open message
on the "jupyter.widget.control" comm channel is received
Returns True if the object has a trait with the specified name.
Hold syncing any state until the outermost context manager exits
Context manager for bundling trait change notifica-
tions and cross validation.
Called when a property has changed.
Setup a handler to be called when a trait changes.
(Un)Register a custom msg receive callback.
(Un)Register a callback to handle text submission.
DEPRECATED: Setup a handler to be called when a trait changes.
Registers a callback to be called when a widget is constructed.
Open a comm to the frontend if one isn't already open.
Removes a class from the top level element of the widget.
Sends a custom msg to the widget model in the frontend.
Sends the widget state, or a piece of it, to the front- end, if it exists.
Called when a state is received from the front-end.
Forcibly sets trait attribute, including read-only attributes.
This is called <b>before</b> selfinit is called.
Return a trait's default value or a dictionary of them
Get a dict of all the event handlers of this class.
Returns True if the specified trait has a value.
Get metadata values for trait by key.
Get a list of all the names of this class' traits.
A dict of trait names and their values.
Get a dict of all the traits of this class.
Remove a trait change handler.
Remove trait change handlers of any type for the specified name.

## **Attributes**

comm	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

## disabled

Enable or disable user changes

## continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

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

Public constructor

## on\_submit(callback, remove=False)

(Un)Register a callback to handle text submission.

Triggered when the user clicks enter.

## **Parameters**

- callback (callable) Will be called with exactly one argument: the Widget instance
- remove (bool (optional)) Whether to unregister the callback

# \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow$ list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# classmethod class\_traits(\*\*metadata: Any) $\rightarrow$ dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

## comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

## description\_allow\_html

Accept HTML in the description.

### property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

### focus()

Focus on the widget.

### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

 $\textbf{key} \, (unicode \ or \ iterable \ (optional)) - A \ single \ property's \ name \ or \ iterable \ of \ property \ names \ to \ get.$ 

#### **Returns**

- **state** (dict of states)
- metadata (dict) metadata for each field: {key: metadata}

```
get_view_spec()
```

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

## layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

## open()

Open a comm to the frontend if one isn't already open.

### placeholder

Placeholder text to display when nothing has been typed

## remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (list of binary buffers) Binary buffers to send with message

# send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

#### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

#### setup\_instance(\*\*kwargs: Any) $\rightarrow$ None

This is called **before** self.\_\_init\_\_ is called.

### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (*str* (*default: None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

## **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SampleSeriesWidget.VBox

```
class AFL.automation.prepare.SampleSeriesWidget.VBox(**kwargs: Any)
    Bases: Box
```

Displays multiple widgets vertically using the flexible box model.

### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '. Applies a predefined style to the box. Defaults to '', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Vertical Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.VBox([title_widget, slider])
```

```
__init__(children=(), **kwargs)
```

Public constructor

## **Methods**

init([children])	Public constructor
<pre>add_class(className)</pre>	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
	widget.

continues on next page

Table 49 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
    Object disposal
__init__(children=(), **kwargs)
    Public constructor
```

add\_class(className)
 Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

## blur()

Blur the widget.

## box\_style

Use a predefined styling for the box.

#### children

List of widget children

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

## **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

### comm

A trait which allows any value.

## property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

## hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

#### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

## tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

## **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

# **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
    widgets = {}
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget(deck)
    __init__(deck)
    apply_protocol_order_cb(event)
    apply_protocol_order()
    update_protocol_order_preview_cb(event)
    make_protocol()
    make_catch_protocol()
    submit_cb(event)
    build_label(index)
    example_label_cb(event)
    make_all_labels_cb(event)
    make_all_labels()
    sync_to_prepare_cb(event)
    reset_uuid_cb(event)
    update_sort_order_cb(event, direction)
    make_mixing_wells()
    update_mixing_well_preview_cb(event)
    submit_mixing_wells_cb(event)
    start()
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Model(deck)
     __init__(deck)
    adjust_protocol_order(mix order)
```

```
\textbf{class} \ \texttt{AFL}. automation. prepare. Sample Series \texttt{Widget}. \textbf{Sample Series} \textbf{Widget}\_\textbf{View}
```

```
__init__()
make_component_grid(components, nsamples)
make_pipette_params(name)
make_mixing_wells()
start(components, nsamples, stock_names)
```

# AFL.automation.prepare.StockBuilderWidget

#### **Functions**

sqrt(x, /)

Return the square root of x.

# AFL.automation.prepare.StockBuilderWidget.sqrt

```
AFL.automation.prepare.StockBuilderWidget.sqrt(x,/)
Return the square root of x.
```

# Classes

```
StockBuilderWidget_Model(deck)
StockBuilderWidget_View()
```

# AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget

# **Methods**

```
_init__(deck)
 add_stocks_to_deck()
 analyze_stocks_cb(event)
 get_stock_objects()
 get_stock_values()
 load_cb(event)
 make_stock_cb(event)
 remove_stock_cb(event)
 save_cb(event[, pkl])
 set_units_cb(event, units)
 start()
 update_location_check_cb(event)
__init__(deck)
analyze_stocks_cb(event)
get_stock_objects()
add_stocks_to_deck()
get_stock_values()
save_cb(event, pkl=True)
load_cb(event)
update_location_check_cb(event)
make_stock_cb(event)
remove_stock_cb(event)
set_units_cb(event, units)
start()
```

# AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget\_Model

#### **Methods**

```
__init__(deck)

add_stocks_to_deck(all_stocks_dict)

to_stock_objects(all_stocks_dict)

__init__(deck)

add_stocks_to_deck(all_stocks_dict)
```

# AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget\_View

to\_stock\_objects(all\_stocks\_dict)

# **Methods**

```
__init__()

make_stock_tab(stock_name, components)

start()

__init__()

make_stock_tab(stock_name, components)

start()

class AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget(deck)

__init__(deck)

analyze_stocks_cb(event)
```

```
get_stock_objects()
    add_stocks_to_deck()
    get_stock_values()
    save_cb(event, pkl=True)
    load_cb(event)
    update_location_check_cb(event)
    make_stock_cb(event)
    remove_stock_cb(event)
    set_units_cb(event, units)
    start()
class AFL.automation.prepare.StockBuilderWidget_Model(deck)
    __init__(deck)
    add_stocks_to_deck(all_stocks_dict)
    to_stock_objects(all_stocks_dict)
class AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget_View
    __init__()
    make_stock_tab(stock_name, components)
    start()
```

# AFL.automation.prepare.SweepBuilderWidget

## **Functions**

sqrt(x,/)

Return the square root of x.

# AFL.automation.prepare.SweepBuilderWidget.sqrt

```
AFL.automation.prepare.SweepBuilderWidget.sqrt(x,/)
Return the square root of x.
```

# Classes

Button(**kwargs)	Button widget.
Checkbox(**kwargs)	Displays a boolean <i>value</i> in the form of a checkbox.
· 6 /	± •
HBox(**kwargs)	Displays multiple widgets horizontally using the flexible
	box model.
Label(**kwargs)	Label widget.
Layout(**kwargs)	Layout specification
SweepBuilderWidget(deck)	
SweepBuilderWidget_Model(deck)	
SweepBuilderWidget_View()	
Text(**kwargs)	Single line textbox widget.
VBox(**kwargs)	Displays multiple widgets vertically using the flexible
( )	box model.

# AFL.automation.prepare.SweepBuilderWidget.Button

class AFL.automation.prepare.SweepBuilderWidget.Button(\*\*kwargs: Any)

Bases: DOMWidget, CoreWidget

Button widget.

This widget has an *on\_click* method that allows you to listen for the user clicking on the button. The click event itself is stateless.

# **Parameters**

- **description** (*str*) description displayed on the button
- icon (str) font-awesome icon names, without the 'fa-' prefix
- **disabled** (*bool*) whether user interaction is enabled

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

Public constructor

# **Methods**

init(**kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not
	a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
click()	Programmatically trigger a click event.

continues on next page

Table 50 – continued from previous page

	ed from previous page
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
<pre>notify_change(change)</pre>	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_click(callback[, remove])	Register a callback to execute when the button is clicked.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the frontend, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

## **Attributes**

button_style	Use a predefined styling for the button.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Button label.
disabled	Enable or disable user changes.
icon	Font-awesome icon names, without the 'fa-' prefix.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

# description

Button label.

#### disabled

Enable or disable user changes.

# icon

Font-awesome icon names, without the 'fa-' prefix.

#### button\_style

Use a predefined styling for the button.

# style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

```
__init__(**kwargs)
```

Public constructor

# on\_click(callback, remove=False)

Register a callback to execute when the button is clicked.

The callback will be called with one argument, the clicked button widget instance.

# **Parameters**

**remove** (bool (optional)) – Set to true to remove the callback from the list of callbacks.

# click()

Programmatically trigger a click event.

This will call the callbacks registered to the clicked button widget instance.

## \_\_del\_\_()

Object disposal

### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

## add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

## **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

## **classmethod class\_own\_traits**(\*\**metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow$ list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata:* Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

# classmethod close\_all()

#### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### **Returns**

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

#### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

# remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

# tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str \mid None = None) \rightarrow dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) \rightarrow dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.SweepBuilderWidget.Checkbox

class AFL.automation.prepare.SweepBuilderWidget.Checkbox(\*\*kwargs: Any)

Bases: \_Bool

Displays a boolean value in the form of a checkbox.

#### **Parameters**

- value ({True, False}) value of the checkbox: True-checked, False-unchecked
- **description** (*str*) description displayed next to the checkbox
- **indent** ({*True*, *False*}) indent the control to align with other controls with a description. The style.description\_width attribute controls this width for consistence with other controls.

```
__init__(value=None, **kwargs)
Public constructor
```

## **Methods**

init([value])	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.

continues on next page

Table 51 – continued from previous page

Table 51 – Continue	ed from previous page
<pre>get_interact_value()</pre>	Return the value for this widget which should be
	passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
<pre>handle_control_comm_opened(comm, msg)</pre>	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
has_trait(name)	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
	manager exits
<pre>hold_trait_notifications()</pre>	Context manager for bundling trait change notifica-
	tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a
	trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is con-
	structed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the
	widget.
<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the front-
	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
	end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only at-
	tributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the spec-
	ified name.

# **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes.
indent	Indent the control to align with other controls with a description.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
style	Styling customizations
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	Bool value
widget_types	
widgets	

# indent

Indent the control to align with other controls with a description.

## style

Styling customizations

# \_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

# classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated:: 8.0.0

Use tooltip attribute instead.

### disabled

Enable or disable user changes.

#### focus()

Focus on the widget.

# get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

#### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

# **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

# remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

# tabbable

Is widget tabbable?

### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str \mid None = None) \rightarrow dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) \rightarrow dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = str | st
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

Bool value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.SweepBuilderWidget.HBox

```
class AFL.automation.prepare.SweepBuilderWidget.HBox(**kwargs: Any)
```

Bases: Box

Displays multiple widgets horizontally using the flexible box model.

## **Parameters**

Public constructor

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '. Applies a predefined style to the box. Defaults to ', which applies no pre-defined style.

## **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Horizontal Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.HBox([title_widget, slider])

__init__(children=(), **kwargs)
```

# Methods

init([children])		
add_traits(**traits)         Dynamically add trait attributes to the Widget.           blur()         Blur the widget.           class_own_trait_events(name)         Get a dict of all event handlers defined on this class, not a parent.           class_trait_names(**metadata)         Get a dict of all the traitlets defined on this class, not a parent.           class_trait_names(**metadata)         Get a dict of all the traits of this class' traits.           class_trait_names(**metadata)         Get a dict of all the traits of this class.           close()         Close method.           focus on the widget.         Returns the full state for a widget manager for embedding.           get_manager_state([drop_defaults, widgets])         Gets the widget state, or a piece of it.           get_view_spec()         Gets the widget state, or a piece of it.           handle_comm_opened(comm, msg)         Static method, called when a widget is constructed.           handle_comm_opened(comm, msg)         Class method, called when a widget is constructed.           handle_comm_opened(comm, msg)         Returns True if the object has a trait with the specified name.           hold_sync()         Hold syncing any state until the outermost context manager exits           hold_sync()         Context manager for bundling trait change notifications and cross validation.           nom.msg(calback], remove)         Context manager for bundling trait change notifications and cross vali		
Blur the widget.   Class_own_trait_events(name)   Get a dict of all event handlers defined on this class, not a parent.   Class_trait_names(**metadata)   Get a dict of all the traitlets defined on this class, not a parent.   Class_trait_names(**metadata)   Get a dict of all the traitlets defined on this class, not a parent.   Get a dict of all the traitlets defined on this class, not a parent.   Get a dict of all the traitlets defined on this class. class_traits(**metadata)   Get a dict of all the traits of this class. close()   Close method.   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. close   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the traits of this class. class   Get a dict of all the vent handlers of this class. class   Get a dict of all the vent handlers of this class. class   Get a dict of all the vent handlers of this class. class   Get a dict of all the vent handlers of this class. class   Get a dict of all the vent handlers of this class. class   Get a dict of all the vent handlers of the miscale   Get a dict of all the vent handlers of the miscale   Get a dict of all the vent handlers of the miscale   Get a dict of all the vent handlers of them   Get a		
class_own_trait_events(name)       Get a dict of all event handlers defined on this class, not a parent.         class_trait_names(**metadata)       Get a dict of all the traitlets defined on this class, not a parent.         class_trait_names(**metadata)       Get a list of all the names of this class' traits.         close()       Get a list of all the names of this class'.         close()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_view_spec()       Gets the widget state, or a piece of it.         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler(, names, type))       Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Cunker jets a custom msg receive callback.		
class_own_traits(**metadata) class_trait_names(**metadata) class_trait_names(**metadata) class_traits(**metadata) class_traits(**metadata) close() close close() close_all()  focus() get_manager_state([drop_defaults, widgets]) get_state([key, drop_defaults]) get_view_spec()  handle_comm_opened(comm, msg) handle_control_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg) con_trait_name)  hold_sync() hold_trait_notifications() hold_trait_notifications() hold_trait_notifications() con_msg(callback[, remove]) on_msg(callback[, remove]) on_msg(callback[, remove]) on_widget_constructed(callback) pen()  on_widget_constructed(callback) send_content[, buffers]) send_state([key]) send_state([hamler]) Se	V	e e e e e e e e e e e e e e e e e e e
a parent.   Class_trait_names(**metadata)   Get a list of all the names of this class' traits.   Class_traits(**metadata)   Get a dict of all the traits of this class.   Close()   Close_all()	<pre>class_own_trait_events(name)</pre>	
class_traits(**metadata)	<pre>class_own_traits(**metadata)</pre>	
close()       Close method.         focus()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for embedding         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler, names, type))       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       On_trait_change([handler, name, remove])         on_widget_constructed(callback)       Registers a custom msg receive callback.         open()       Open a comm to the frontend if one isn't already open.         remove_class(className)       Removes a class from the top level element of the widget.         send_state([key])       Sends a custom msg to the widget model in the frontend, if it exists.         set_state(	<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
close_all()       Focus on the widget.         get_manager_state([drop_defaults, widgets])       Focus on the widget.         get_state([key, drop_defaults])       Gets the widget state for a widget manager for embedding         get_view_spec()       Static method, called when a widget is constructed.         handle_comm_opened(comm, msg)       Static method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_midget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open.         remove_class(className)       Removes a class from the top level element of the widget.         send_state([key])       Sends a custom msg to the widget model in the frontend, if it exists.         set_state(sync_data)       Sends the widget state, or a piece of it, to the fr	<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
Focus on the widget.   Returns the full state for a widget manager for embedding   Gets the widget state, or a piece of it.	close()	Close method.
get_state([key, drop_defaults])         Returns the full state for a widget manager for embedding           get_state([key, drop_defaults])         Gets the widget state, or a piece of it.           get_view_spec()         Static method, called when a widget is constructed.           handle_comm_opened(comm, msg)         Static method, called when a widget is constructed.           handle_control_comm_opened(comm, msg)         Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received           has_trait(name)         Returns True if the object has a trait with the specified name.           hold_sync()         Hold syncing any state until the outermost context manager exits           hold_trait_notifications()         Context manager for bundling trait change notifications and cross validation.           notify_change(change)         Called when a property has changed.           observe(handler, names, type])         Setup a handler to be called when a trait changes.           on_trait_change([handler, name, remove])         DEPRECATED: Setup a handler to be called when a trait changes.           on_widget_constructed(callback)         Registers a callback to be called when a widget is constructed.           open()         Open a comm to the frontend if one isn't already open.           remove_class(className)         Removes a class from the top level element of the widget.           send_state([key])         Sends the widget state, or a piece of it, to	close_all()	
ding  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  has_trait(name)  hold_sync()  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Called when a property has changed.  Called when a property has changed.  Observe(handler[, names, type])  on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a widget is constructed.  Context manager axist  Context manager for bundling trait change notifications and cross validation.  Despect of the property has changed.  Setup a handler to be called when a trait changes.  On_msg(callback], remove])  On_trait_change([handler, name, remove])  Deprecated is a custom msg receive callback.  Registers a callback to be called when a widget is constructed.  Open()  Open a comm to the frontend if one isn't already open.  remove_class(className)  Removes a class from the top level element of the widget.  send_state([key])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  called when a state is received from the front-end.  set_state(sync_data)  Setup instance(**kwargs)  This is called before selfinit is called.  Return a trait's default value or a dictionary of them trait_devents([name])  Get a dict of all the event handlers of this class.	focus()	Focus on the widget.
handle_comm_opened(comm, msg)   Static method, called when a widget is constructed.   handle_control_comm_opened(comm, msg)   Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received   Returns True if the object has a trait with the specified name.   hold_sync()   Hold syncing any state until the outermost context manager exits   hold_trait_notifications()   Context manager for bundling trait change notifications and cross validation.   hotify_change(change)   Called when a property has changed.   hoserve(handler[, names, type])   Setup a handler to be called when a trait changes.   hold_trait_notifications()   Culled when a property has changed.   hotify_change(change)   Called when a property has changed.   hotify_change(change)   Called when a property has changed.   hotify_change(change)   DEPRECATED: Setup a handler to be called when a trait changes.   hotify_change(landler, name, remove])   DEPRECATED: Setup a handler to be called when a trait changes.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback.   hotify_change(landler, name, remove])   DEPRECATED: Setup a handler to be called when a trait changes.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback to be called when a widget is constructed.   hotify_change(landler, name, remove])   Deprecate a custom msg receive callback to be called when a widget is constructed.   hotify_change(landler, name, remove)   Deprecate a custom msg receive callback to be called when a widget is constructed.   hotify_change(landler, name, remove)   Deprecate a custom msg receive callback to be called when a widget is constructed.   hotify_change(landler, name, remove)   Deprecate a custom msg receive callback to be c	<pre>get_manager_state([drop_defaults, widgets])</pre>	
handle_comm_opened(comm, msg)       Static method, called when a widget is constructed.         handle_control_comm_opened(comm, msg)       Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received         has_trait(name)       Returns True if the object has a trait with the specified name.         hold_sync()       Hold syncing any state until the outermost context manager exits         hold_trait_notifications()       Context manager for bundling trait change notifications and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait changes.         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.         open()       Open a comm to the frontend if one isn't already open.         remove_class(className)       Removes a class from the top level element of the widget.         send_state([key])       Sends the widget state, or a piece of it, to the frontend.         set_state(sync_data)       Sends the widget state, or a piece of it, to the frontend.         set_trait(name, value)       Forcibly sets trait attribute, including read-only attributes.		Gets the widget state, or a piece of it.
handle_control_comm_opened(comm, msg)Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is receivedhas_trait(name)Returns True if the object has a trait with the specified name.hold_sync()Hold syncing any state until the outermost context manager exitshold_trait_notifications()Context manager for bundling trait change notifications and cross validation.notify_change(change)Called when a property has changed.observe(handler[, names, type])Setup a handler to be called when a trait changes.on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send_state([key])Sends a custom msg to the widget model in the frontend, if it exists.set_state(sync_data)Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit_ is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themGet a dict of all the event handlers of this class.	<pre>get_view_spec()</pre>	
on the "jupyter.widget.control" comm channel is received  has_trait(name)  Returns True if the object has a trait with the specified name.  hold_sync()  Hold syncing any state until the outermost context manager exits  hold_trait_notifications()  Context manager for bundling trait change notifications and cross validation.  notify_change(change)  observe(handler[, names, type])  Setup a handler to be called when a trait changes.  on_msg(callback[, remove])  (Un)Register a custom msg receive callback.  on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait changes.  on_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  open()  Open a comm to the frontend if one isn't already open.  remove_class(className)  Removes a class from the top level element of the widget.  send_state([key])  Sends a custom msg to the widget model in the frontend.  set_state(sync_data)  Sends the widget state, or a piece of it, to the frontend, if it exists.  set_state(sync_data)  Sends the widget state is received from the frontend.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit_ is called.  trait_defaults(*names, **metadata)  Get a dict of all the event handlers of this class.	<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
has_trait(name)Returns True if the object has a trait with the specified name.hold_sync()Hold syncing any state until the outermost context manager exitshold_trait_notifications()Context manager for bundling trait change notifications and cross validation.notify_change(change)Called when a property has changed.observe(handler[, names, type])Setup a handler to be called when a trait changes.on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the frontend.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of them trait_events([name])	handle_control_comm_opened(comm, msg)	on the "jupyter.widget.control" comm channel is re-
hold_sync()Hold syncing any state until the outermost context manager exitshold_trait_notifications()Context manager for bundling trait change notifications and cross validation.notify_change(change)Called when a property has changed.observe(handler[, names, type])Setup a handler to be called when a trait changes.on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Sends the widget state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*name, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	has_trait(name)	Returns True if the object has a trait with the specified
tions and cross validation.  notify_change(change)	hold_sync()	
observe(handler[, names, type])Setup a handler to be called when a trait changes.on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the frontend.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	hold_trait_notifications()	
on_msg(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the frontend.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit_ is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	notify_change(change)	Called when a property has changed.
on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a trait changes.on_widget_constructed(callback)Registers a callback to be called when a widget is constructed.open()Open a comm to the frontend if one isn't already open.remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit_ is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	<pre>observe(handler[, names, type])</pre>	Setup a handler to be called when a trait changes.
trait changes.  on_widget_constructed(callback)  Registers a callback to be called when a widget is constructed.  open()  open a comm to the frontend if one isn't already open.  Removes a class from the top level element of the widget.  send(content[, buffers])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  set_state(sync_data)  Set_trait(name, value)  Called when a state is received from the front-end.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them trait_events([name])  Get a dict of all the event handlers of this class.	<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
structed.  open()  remove_class(className)  Removes a class from the top level element of the widget.  send(content[, buffers])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  set_state(sync_data)  Set_trait(name, value)  Called when a state is received from the frontend.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them trait_events([name])  Get a dict of all the event handlers of this class.	<pre>on_trait_change([handler, name, remove])</pre>	
remove_class(className)Removes a class from the top level element of the widget.send(content[, buffers])Sends a custom msg to the widget model in the frontend.send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the frontend.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	on_widget_constructed(callback)	
widget.  send(content[, buffers])  Sends a custom msg to the widget model in the frontend.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  set_state(sync_data)  Set_trait(name, value)  Called when a state is received from the frontend.  set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.	open()	Open a comm to the frontend if one isn't already open.
end.  send_state([key])  Sends the widget state, or a piece of it, to the frontend, if it exists.  set_state(sync_data)  Set_trait(name, value)  Forcibly sets trait attribute, including read-only attributes.  setup_instance(**kwargs)  This is called before selfinit is called.  trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.	remove_class(className)	•
send_state([key])Sends the widget state, or a piece of it, to the frontend, if it exists.set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	<pre>send(content[, buffers])</pre>	Sends a custom msg to the widget model in the front-
set_state(sync_data)Called when a state is received from the front-end.set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	<pre>send_state([key])</pre>	Sends the widget state, or a piece of it, to the front-
set_trait(name, value)Forcibly sets trait attribute, including read-only attributes.setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	set_state(sync_data)	
setup_instance(**kwargs)This is called before selfinit is called.trait_defaults(*names, **metadata)Return a trait's default value or a dictionary of themtrait_events([name])Get a dict of all the event handlers of this class.	· •	·
trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of them  trait_events([name])  Get a dict of all the event handlers of this class.	setup_instance(**kwargs)	
trait_events([name]) Get a dict of all the event handlers of this class.	- , , , , ,	
		·
trait_nas_value(name) Returns True if the specified trait has a value.	trait_has_value(name)	Returns True if the specified trait has a value.

continues on next page

Table 52 – continued from previous page

trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

# \_\_del\_\_()

Object disposal

\_\_init\_\_(children=(), \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

# box\_style

Use a predefined styling for the box.

# children

List of widget children

### classmethod class\_own\_trait\_events(name: str) $\rightarrow$ dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

# classmethod class\_trait\_names(\*\*metadata: Any) $\rightarrow list[str]$

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

# property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

# Returns

#### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

# get\_view\_spec()

## static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

# hold\_sync()

Hold syncing any state until the outermost context manager exits

### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

## keys

The traits which are synced.

### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### loa

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

## notify\_change(change)

Called when a property has changed.

**observe**(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change')  $\rightarrow$  None

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

## **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- **name** (*list*, *str*, *None*) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- remove (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

# send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

## send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

# set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

### tabbable

Is widget tabbable?

## tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

## Notes

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# Return type

A dict of trait names and their values.

# **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

# **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

# AFL.automation.prepare.SweepBuilderWidget.Label

```
\textbf{class} \ \texttt{AFL}. \textbf{automation.prepare.SweepBuilderWidget.Label(} \textbf{**kwargs: } \textit{Any} \textbf{)}
```

Bases: \_String

Label widget.

It also renders math inside the string *value* as Latex (requires \$ \$ or \$\$ \$\$ and similar latex tags).

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

### **Methods**

init([value])	Public constructor
<pre>add_class(className)</pre>	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_interact_value()</pre>	Return the value for this widget which should be passed to interactive functions.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
	continues on next page

continues on next page

Table 53 – continued from previous page

	od nom proviodo pago
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
<pre>on_trait_change([handler, name, remove])</pre>	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified
	class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been
	typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

## style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## \_\_del\_\_()

Object disposal

\_\_init\_\_(value=None, \*\*kwargs)

Public constructor

# add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

# add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

# **classmethod class\_own\_trait\_events**(name: str) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

### classmethod class\_own\_traits(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

### **classmethod class\_trait\_names(\*\****metadata: Any*) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

# **classmethod class\_traits**(\*\**metadata*: *Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

## classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

# description

Description of the control.

### description\_allow\_html

Accept HTML in the description.

# property description\_tooltip

The tooltip information. .. deprecated:: 8.0.0

Use tooltip attribute instead.

#### focus()

Focus on the widget.

#### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

## get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (*unicode* or *iterable* (*optional*)) – A single property's name or iterable of property names to get.

#### Returns

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

## get\_view\_spec()

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

### classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

# layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

#### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

```
notify_change(change)
```

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

# **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.

• remove (bool) – If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

## placeholder

Placeholder text to display when nothing has been typed

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

### send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

### send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

#### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

# $trait_defaults(*names: str, **metadata: Any) \rightarrow dict[str, Any] | Sentinel$

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.SweepBuilderWidget.Layout

```
class AFL.automation.prepare.SweepBuilderWidget.Layout(**kwargs: Any)
```

Bases: Widget

Layout specification

Defines a layout that can be expressed using CSS. Supports a subset of https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

When a property is also accessible via a shorthand property, we only expose the shorthand.

For example: - flex-grow, flex-shrink and flex-basis are bound to flex. - flex-wrap and flex-direction are bound to flex-flow. - margin-[top/bottom/left/right] values are bound to margin, etc.

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

Public constructor

# Methods

init(**kwargs)	Public constructor
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not
	a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
v	
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
g()	
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message
	on the "jupyter.widget.control" comm channel is re-
	ceived
has_trait(name)	Returns True if the object has a trait with the specified
	name.
hold_sync()	Hold syncing any state until the outermost context
1014_5/116()	manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica-
	tions and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a
on_trare_enange([nanoior, name, remove])	trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is con-
on_wrage c_constructed (canoack)	structed.
open()	Open a comm to the frontend if one isn't already open.
send(content[, buffers])	Sends a custom msg to the widget model in the front-
Scha (content)	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
Sena_State([Rey])	end, if it exists.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only at-
Set_trart(name, value)	tributes.
setup_instance(**kwargs)	This is called <b>before</b> selfinit is called.
trait_defaults(*names, **metadata)	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
trait_mas_varue(name)  trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
crarc_metauata(naimame, key[, defauit])	Oct inclaudia values for traft by key.

continues on next page

Table 54 – continued from previous page

trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

# **Attributes**

7.4	The state of the s
align_content	The align-content CSS attribute.
align_items	The align-items CSS attribute.
align_self	The align-self CSS attribute.
border	border property getter.
border_bottom	The border bottom CSS attribute.
border_left	The border left CSS attribute.
border_right	The border right CSS attribute.
border_top	The border top CSS attribute.
bottom	The bottom CSS attribute.
COMM	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross
	validation lock set to True.
display	The display CSS attribute.
flex	The flex CSS attribute.
flex_flow	The flex-flow CSS attribute.
grid_area	The grid-area CSS attribute.
grid_auto_columns	The grid-auto-columns CSS attribute.
grid_auto_flow	The grid-auto-flow CSS attribute.
grid_auto_rows	The grid-auto-rows CSS attribute.
grid_column	The grid-column CSS attribute.
grid_gap	The grid-gap CSS attribute.
grid_row	The grid-row CSS attribute.
grid_template_areas	The grid-template-areas CSS attribute.
grid_template_columns	The grid-template-columns CSS attribute.
grid_template_rows	The grid-template-rows CSS attribute.
height	The height CSS attribute.
justify_content	The justify-content CSS attribute.
justify_items	The justify-items CSS attribute.
keys	The traits which are synced.
left	The left CSS attribute.
log	A trait whose value must be an instance of a specified
	class.
margin	The margin CSS attribute.
max_height	The max-height CSS attribute.
max_width	The max-width CSS attribute.
min_height	The min-height CSS attribute.
min_width	The min-width CSS attribute.
model_id	Gets the model id of this widget.
object_fit	The object-fit CSS attribute.
object_position	The object-position CSS attribute.
order	The order CSS attribute.
	continues on next page

continues on next page

Table 55 – continued from previous page

overflow	The overflow CSS attribute.
padding	The padding CSS attribute.
right	The right CSS attribute.
top	The top CSS attribute.
visibility	The visibility CSS attribute.
widget_types	
widgets	
width	The width CSS attribute.

# align\_content

The align-content CSS attribute.

## align\_items

The align-items CSS attribute.

# align\_self

The align-self CSS attribute.

## border\_top

The border top CSS attribute.

## border\_right

The border right CSS attribute.

### border\_bottom

The border bottom CSS attribute.

## border\_left

The border left CSS attribute.

## bottom

The bottom CSS attribute.

### display

The display CSS attribute.

### flex

The flex CSS attribute.

## flex\_flow

The flex-flow CSS attribute.

## height

The height CSS attribute.

## justify\_content

The justify-content CSS attribute.

# justify\_items

The justify-items CSS attribute.

### left

The left CSS attribute.

### margin

The margin CSS attribute.

# max\_height

The max-height CSS attribute.

## max\_width

The max-width CSS attribute.

## min\_height

The min-height CSS attribute.

## min\_width

The min-width CSS attribute.

### overflow

The overflow CSS attribute.

## order

The order CSS attribute.

## padding

The padding CSS attribute.

## right

The right CSS attribute.

#### top

The top CSS attribute.

# visibility

The visibility CSS attribute.

## width

The width CSS attribute.

# object\_fit

The object-fit CSS attribute.

## object\_position

The object-position CSS attribute.

## grid\_auto\_columns

The grid-auto-columns CSS attribute.

## grid\_auto\_flow

The grid-auto-flow CSS attribute.

## grid\_auto\_rows

The grid-auto-rows CSS attribute.

### grid\_gap

The grid-gap CSS attribute.

# grid\_template\_rows

The grid-template-rows CSS attribute.

#### grid\_template\_columns

The grid-template-columns CSS attribute.

### grid\_template\_areas

The grid-template-areas CSS attribute.

#### grid\_row

The grid-row CSS attribute.

#### grid\_column

The grid-column CSS attribute.

#### grid\_area

The grid-area CSS attribute.

```
__init__(**kwargs)
```

Public constructor

### property border

border property getter. Return the common value of all side borders if they are identical. Otherwise return None.

```
__del__()
```

Object disposal

#### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

```
classmethod class_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the traits() method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

#### comm

A trait which allows any value.

#### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### Returns

- state (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

# static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

## classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

## hold\_sync()

Hold syncing any state until the outermost context manager exits

#### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

## property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

#### on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

## **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (*callable*, *None*) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- **name** (*list*, *str*, *None*) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

## static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup\_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self.\_\_init\_\_ is called.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

#### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) \rightarrow bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait\_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) \rightarrow dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## Return type

A dict of trait names and their values.

#### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = traitlets.All, type: traitlets.All, type:
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widget.WidgetRegistry object>
widgets = {}
```

## AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget

#### **Methods**

```
__init__(deck)

calc_sweep_cb(click)

get_deck()

get_sweep_data()

plot_binary_cb(click)

plot_ternary_cb(click)

start()

update_component_row_cb(event)

validate_sweep_cb(click)

__init__(deck)

plot_binary_cb(click)
```

```
plot_ternary_cb(click)
     calc_sweep_cb(click)
     validate_sweep_cb(click)
     get_sweep_data()
     get_deck()
     update_component_row_cb(event)
     start()
AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget_Model
class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget_Model(deck)
     Bases: object
     __init__(deck)
     Methods
      __init__(deck)
      calc_sweep(sweep_dict, progress)
      validate_sweep(tolerance[, progress])
     __init__(deck)
     validate_sweep(tolerance, progress=None)
     calc_sweep(sweep_dict, progress)
AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget_View
class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget_View
     Bases: object
     __init__()
```

## **Methods**

```
__init__()

make_binary_plot(component_names)

make_stock_grid(component_names)

make_ternary_plot(component_names)

start(component_names)
```

```
__init__()
make_stock_grid(component_names)
make_ternary_plot(component_names)
make_binary_plot(component_names)
start(component_names)
```

# AFL.automation.prepare.SweepBuilderWidget.Text

### **Methods**

init(*args, **kwargs)	Public constructor
add_class(className)	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class,
	not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not
	a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.

continues on next page

Table 56 – continued from previous page

get_interact_value()       Return the value for this widget which should passed to interactive functions.         get_manager_state([drop_defaults, widgets])       Returns the full state for a widget manager for ending ding         get_state([key, drop_defaults])       Gets the widget state, or a piece of it.         get_view_spec()       Static method, called when a widget is constructed that the commopened common the "jupyter.widget.control" common common the "jupyter.widget.control" common common thannel control common thannel control common that the special control common that the common than the common than the common than the common that the common common that the common that the common than the common that the common than the common that the common than the common than the common than the common that the common that the common that the common than the common that the common than the common that the common that the common that the common than the common than the common than the common that the common than the common than the common than the common that the common than the common	eted. essage is re- cified
ding  get_state([key, drop_defaults])  fets the widget state, or a piece of it.  get_view_spec()  handle_comm_opened(comm, msg)  handle_control_comm_opened(comm, msg)  Class method, called when a widget is construct on the "jupyter.widget.control" comm channel ceived  has_trait(name)  Returns True if the object has a trait with the speciame.  hold_sync()  Hold syncing any state until the outermost comanager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  observe(handler[, names, type])  on_msg(callback[, remove])  on_submit(callback[, remove])  on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is set.	ested. essage is re- cified ontext
get_view_spec()         handle_comm_opened(comm, msg)       Static method, called when a widget is construct than defended from the "jupyter.widget.control" comm channel ceived         has_trait(name)       Returns True if the object has a trait with the speciane.         hold_sync()       Hold syncing any state until the outermost commanager exits         hold_trait_notifications()       Context manager for bundling trait change not tions and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait change on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       (Un)Register a callback to handle text submission on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a widget is constructed (callback)         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.	essage is re- cified ontext
get_view_spec()         handle_comm_opened(comm, msg)       Static method, called when a widget is construct than defended from the "jupyter.widget.control" comm channel ceived         has_trait(name)       Returns True if the object has a trait with the specians.         hold_sync()       Hold syncing any state until the outermost commanager exits         hold_trait_notifications()       Context manager for bundling trait change not tions and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait change on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       (Un)Register a callback to handle text submission on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called when a widget is constructed (callback)         on_widget_constructed(callback)       Registers a callback to be called when a widget is constructed.	essage is re- cified ontext
handle_comm_opened(comm, msg)       Static method, called when a widget is constructed handle_control_comm_opened(comm, msg)       Class method, called when the comm-open menon the "jupyter.widget.control" comm channel ceived         has_trait(name)       Returns True if the object has a trait with the speciane.         hold_sync()       Hold syncing any state until the outermost commanager exits         hold_trait_notifications()       Context manager for bundling trait change not tions and cross validation.         notify_change(change)       Called when a property has changed.         observe(handler[, names, type])       Setup a handler to be called when a trait change on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       DEPRECATED: Setup a handler to be called when a widget is constructed callback)         Registers a callback to be called when a widget is constructed.	essage is re- cified ontext
handle_control_comm_opened(comm, msg)Class method, called when the comm-open menor the "jupyter.widget.control" comm channel ceivedhas_trait(name)Returns True if the object has a trait with the special name.hold_sync()Hold syncing any state until the outermost commanager exitshold_trait_notifications()Context manager for bundling trait change not tions and cross validation.notify_change(change)Called when a property has changed.observe(handler[, names, type])Setup a handler to be called when a trait change on_msg(callback[, remove])on_submit(callback[, remove])(Un)Register a callback to handle text submission_trait_change([handler, name, remove])on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a widget ison_widget_constructed(callback)Registers a callback to be called when a widget is	essage is re- cified ontext
handle_control_comm_opened(comm, msg)Class method, called when the comm-open menor the "jupyter.widget.control" comm channel ceivedhas_trait(name)Returns True if the object has a trait with the special name.hold_sync()Hold syncing any state until the outermost commanager exitshold_trait_notifications()Context manager for bundling trait change not tions and cross validation.notify_change(change)Called when a property has changed.observe(handler[, names, type])Setup a handler to be called when a trait change on_msg(callback[, remove])on_submit(callback[, remove])(Un)Register a custom msg receive callback.on_trait_change([handler, name, remove])DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)Registers a callback to be called when a widget is	essage is re- cified ontext
on the "jupyter.widget.control" comm channel ceived  has_trait(name)  Returns True if the object has a trait with the spect name.  hold_sync()  Hold syncing any state until the outermost commanager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is	is recified
name.  hold_sync()  Hold syncing any state until the outermost commanager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait change on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is	ontext
name.  hold_sync()  Hold syncing any state until the outermost commanager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a trait change on_trait_change([handler, name, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is	ontext
manager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  On_widget_constructed(callback)  manager exits  Context manager for bundling trait change not tions and cross validation.  Called when a property has changed.  Setup a handler to be called when a trait change (Un)Register a custom msg receive callback.  On_submit(callback[, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is	
manager exits  hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  On_widget_constructed(callback)  manager exits  Context manager for bundling trait change not tions and cross validation.  Called when a property has changed.  Setup a handler to be called when a trait change (Un)Register a custom msg receive callback.  On_submit(callback[, remove])  DEPRECATED: Setup a handler to be called when a widget is on_widget_constructed(callback)  Registers a callback to be called when a widget is	
hold_trait_notifications()  Context manager for bundling trait change not tions and cross validation.  notify_change(change)  Observe(handler[, names, type])  On_msg(callback[, remove])  On_submit(callback[, remove])  On_trait_change([handler, name, remove])  On_widget_constructed(callback)  Called when a property has changed.  Setup a handler to be called when a trait change (Un)Register a custom msg receive callback.  On_submit(callback[, remove])  DEPRECATED: Setup a handler to be called when a widget is con_widget_constructed(callback)  Registers a callback to be called when a widget is	
tions and cross validation.  notify_change(change) Called when a property has changed.  observe(handler[, names, type]) Setup a handler to be called when a trait change on_msg(callback[, remove]) (Un)Register a custom msg receive callback.  on_submit(callback[, remove]) (Un)Register a callback to handle text submission on_trait_change([handler, name, remove]) DEPRECATED: Setup a handler to be called we trait changes.  on_widget_constructed(callback) Registers a callback to be called when a widget is	tifica-
observe(handler[, names, type])       Setup a handler to be called when a trait change         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       (Un)Register a callback to handle text submission         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called we trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is	
observe(handler[, names, type])       Setup a handler to be called when a trait change         on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       (Un)Register a callback to handle text submission         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called we trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is	
on_msg(callback[, remove])       (Un)Register a custom msg receive callback.         on_submit(callback[, remove])       (Un)Register a callback to handle text submission.         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called with trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is	es.
on_submit(callback[, remove])       (Un)Register a callback to handle text submission         on_trait_change([handler, name, remove])       DEPRECATED: Setup a handler to be called with trait changes.         on_widget_constructed(callback)       Registers a callback to be called when a widget is	
trait changes. on_widget_constructed(callback)  Registers a callback to be called when a widget is	on.
trait changes. on_widget_constructed(callback)  Registers a callback to be called when a widget is	hen a
	s con-
open() Open a comm to the frontend if one isn't already	open.
remove_class(className)  Removes a class from the top level element of widget.	of the
send(content[, buffers])  Sends a custom msg to the widget model in the fend.	front-
send_state([key]) Sends the widget state, or a piece of it, to the fend, if it exists.	front-
set_state(sync_data) Called when a state is received from the front-e	nd.
set_trait(name, value)  Forcibly sets trait attribute, including read-only tributes.	ly at-
setup_instance(**kwargs) This is called <b>before</b> selfinit is called.	
trait_defaults(*names, **metadata)  Return a trait's default value or a dictionary of t	hem
trait_events([name]) Get a dict of all the event handlers of this class	S.
trait_has_value(name) Returns True if the specified trait has a value.	
trait_metadata(traitname, key[, default]) Get metadata values for trait by key.	
trait_names(**metadata) Get a list of all the names of this class' traits.	
trait_values(**metadata) A dict of trait names and their values.	
traits(**metadata) Get a dict of all the traits of this class.	
unobserve(handler[, names, type]) Remove a trait change handler.	
<i>unobserve_all</i> ([name]) Remove trait change handlers of any type for the ified name.	spec-

## **Attributes**

comm	A trait which allows any value.
continuous_update	Update the value as the user types.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
description	Description of the control.
description_allow_html	Accept HTML in the description.
description_tooltip	The tooltip information deprecated :: 8.0.0 Use tooltip attribute instead.
disabled	Enable or disable user changes
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
placeholder	Placeholder text to display when nothing has been typed
style	An instance trait which coerces a dict to an instance.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
value	String value
widget_types	
widgets	

### disabled

Enable or disable user changes

## continuous\_update

Update the value as the user types. If False, update on submission, e.g., pressing Enter or navigating away.

### style

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

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

Public constructor

## on\_submit(callback, remove=False)

(Un)Register a callback to handle text submission.

Triggered when the user clicks enter.

## **Parameters**

- callback (callable) Will be called with exactly one argument: the Widget instance
- remove (bool (optional)) Whether to unregister the callback

# \_\_del\_\_()

Object disposal

#### add\_class(className)

Adds a class to the top level element of the widget.

Doesn't add the class if it already exists.

#### add\_traits(\*\*traits)

Dynamically add trait attributes to the Widget.

# blur()

Blur the widget.

#### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

# **classmethod class\_own\_traits**(\*\*metadata: Any) → dict[str, TraitType[Any, Any]]

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

## classmethod class\_trait\_names(\*\*metadata: Any) → list[str]

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

### **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

# close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

## description

Description of the control.

#### description\_allow\_html

Accept HTML in the description.

#### property description\_tooltip

The tooltip information. .. deprecated :: 8.0.0

Use tooltip attribute instead.

#### focus()

Focus on the widget.

#### get\_interact\_value()

Return the value for this widget which should be passed to interactive functions. Custom widgets can change this method to process the raw value self.value.

### static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- drop\_defaults when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

#### Returns

```
get_state(key=None, drop_defaults=False)
```

Gets the widget state, or a piece of it.

#### **Parameters**

 $\textbf{key} \, (unicode \ or \ iterable \ (optional)) - A \ single \ property's \ name \ or \ iterable \ of \ property \ names \ to \ get.$ 

#### **Returns**

- **state** (dict of states)
- metadata (dict) metadata for each field: {key: metadata}

```
get_view_spec()
```

### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

### hold\_sync()

Hold syncing any state until the outermost context manager exits

### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

#### keys

The traits which are synced.

#### layout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

#### log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default*: '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

```
on_msg(callback, remove=False)
```

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

### static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

### open()

Open a comm to the frontend if one isn't already open.

#### placeholder

Placeholder text to display when nothing has been typed

### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

## send(content, buffers=None)

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- **buffers** (list of binary buffers) Binary buffers to send with message

# send\_state(key=None)

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### **Parameters**

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

#### set\_state(sync\_data)

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

#### setup\_instance(\*\*kwargs: Any) $\rightarrow$ None

This is called **before** self.\_\_init\_\_ is called.

#### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

#### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

#### **Parameters**

**name** (str (default: None)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

#### Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) \rightarrow list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

#### Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

#### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change')*) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

#### value

String value

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
widgets = {}
```

### AFL.automation.prepare.SweepBuilderWidget.VBox

```
class AFL.automation.prepare.SweepBuilderWidget.VBox(**kwargs: Any)
    Bases: Box
```

Displays multiple widgets vertically using the flexible box model.

#### **Parameters**

- children (iterable of Widget instances) list of widgets to display
- **box\_style** (*str*) one of 'success', 'info', 'warning' or 'danger', or '. Applies a predefined style to the box. Defaults to '', which applies no pre-defined style.

# **Examples**

```
>>> import ipywidgets as widgets
>>> title_widget = widgets.HTML('<em>Vertical Box Example</em>')
>>> slider = widgets.IntSlider()
>>> widgets.VBox([title_widget, slider])
```

 $\verb|\__init__(children=(), **kwargs)|\\$ 

Public constructor

## **Methods**

init([children])	Public constructor
<pre>add_class(className)</pre>	Adds a class to the top level element of the widget.
<pre>add_traits(**traits)</pre>	Dynamically add trait attributes to the Widget.
blur()	Blur the widget.
<pre>class_own_trait_events(name)</pre>	Get a dict of all event handlers defined on this class, not a parent.
<pre>class_own_traits(**metadata)</pre>	Get a dict of all the traitlets defined on this class, not a parent.
<pre>class_trait_names(**metadata)</pre>	Get a list of all the names of this class' traits.
<pre>class_traits(**metadata)</pre>	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
focus()	Focus on the widget.
<pre>get_manager_state([drop_defaults, widgets])</pre>	Returns the full state for a widget manager for embedding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	
<pre>handle_comm_opened(comm, msg)</pre>	Static method, called when a widget is constructed.
handle_control_comm_opened(comm, msg)	Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is re- ceived
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
	widget.

continues on next page

Table 57 – continued from previous page

send(content[, buffers])	Sends a custom msg to the widget model in the frontend.
send_state([key])	Sends the widget state, or a piece of it, to the front- end, if it exists.
<pre>set_state(sync_data)</pre>	Called when a state is received from the front-end.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called <b>before</b> selfinit is called.
<pre>trait_defaults(*names, **metadata)</pre>	Return a trait's default value or a dictionary of them
trait_events([name])	Get a dict of all the event handlers of this class.
trait_has_value(name)	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default])</pre>	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
trait_values(**metadata)	A dict of trait names and their values.
traits(**metadata)	Get a dict of all the traits of this class.
<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

### **Attributes**

box_style	Use a predefined styling for the box.
children	List of widget children
comm	A trait which allows any value.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
keys	The traits which are synced.
layout	An instance trait which coerces a dict to an instance.
log	A trait whose value must be an instance of a specified class.
model_id	Gets the model id of this widget.
tabbable	Is widget tabbable?
tooltip	A tooltip caption.
widget_types	
widgets	

```
__del__()
   Object disposal
__init__(children=(), **kwargs)
   Public constructor
add_class(className)
   Adds a class to the top level element of the widget.
   Doesn't add the class if it already exists.
add_traits(**traits)
```

Dynamically add trait attributes to the Widget.

#### blur()

Blur the widget.

#### box\_style

Use a predefined styling for the box.

### children

List of widget children

#### **classmethod class\_own\_trait\_events**(*name: str*) → dict[str, EventHandler]

Get a dict of all event handlers defined on this class, not a parent.

Works like event\_handlers, except for excluding traits from parents.

```
classmethod class_own_traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traitlets defined on this class, not a parent.

Works like *class\_traits*, except for excluding traits from parents.

```
classmethod class_trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

This method is just like the trait\_names() method, but is unbound.

### **classmethod class\_traits(\*\****metadata: Any*) → dict[str, TraitType[Any, Any]]

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

This method is just like the *traits()* method, but is unbound.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

## close()

Close method.

Closes the underlying comm. When the comm is closed, all of the widget views are automatically removed from the front-end.

### classmethod close\_all()

#### comm

A trait which allows any value.

### property cross\_validation\_lock: Any

A contextmanager for running a block with our cross validation lock set to True.

At the end of the block, the lock's value is restored to its value prior to entering the block.

#### focus()

Focus on the widget.

## static get\_manager\_state(drop\_defaults=False, widgets=None)

Returns the full state for a widget manager for embedding

#### **Parameters**

- **drop\_defaults** when True, it will not include default value
- widgets list with widgets to include in the state (or all widgets when None)

### Returns

### get\_state(key=None, drop\_defaults=False)

Gets the widget state, or a piece of it.

#### **Parameters**

**key** (unicode or iterable (optional)) – A single property's name or iterable of property names to get.

#### **Returns**

- **state** (dict of states)
- **metadata** (*dict*) metadata for each field: {key: metadata}

```
get_view_spec()
```

#### static handle\_comm\_opened(comm, msg)

Static method, called when a widget is constructed.

# classmethod handle\_control\_comm\_opened(comm, msg)

Class method, called when the comm-open message on the "jupyter.widget.control" comm channel is received

```
has\_trait(name: str) \rightarrow bool
```

Returns True if the object has a trait with the specified name.

#### hold\_sync()

Hold syncing any state until the outermost context manager exits

### hold\_trait\_notifications() → Any

Context manager for bundling trait change notifications and cross validation.

Use this when doing multiple trait assignments (init, config), to avoid race conditions in trait notifiers requesting other trait values. All trait notifications will fire after all values have been assigned.

### keys

The traits which are synced.

#### lavout

An instance trait which coerces a dict to an instance.

This lets the instance be specified as a dict, which is used to initialize the instance.

Also, we default to a trivial instance, even if args and kwargs is not specified.

## log

A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

### property model\_id

Gets the model id of this widget.

If a Comm doesn't exist yet, a Comm will be created automagically.

#### notify\_change(change)

Called when a property has changed.

```
observe(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | str = 'change') \rightarrow None
```

Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

#### **Parameters**

- handler (callable) A callable that is called when a trait changes. Its signature should be handler(change), where change is a dictionary. The change dictionary at least holds a 'type' key. \* type: the type of notification. Other keys may be passed depending on the value of 'type'. In the case where type is 'change', we also have the following keys: \* owner: the HasTraits instance \* old: the old value of the modified trait attribute \* new: the new value of the modified trait attribute \* name: the name of the modified trait attribute.
- names (list, str, All) If names is All, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **type** (*str*, *All* (*default:* '*change*')) The type of notification to filter by. If equal to All, then all notifications are passed to the observe handler.

on\_msg(callback, remove=False)

(Un)Register a custom msg receive callback.

#### **Parameters**

• callback (callable) – callback will be passed three arguments when a message arrives:

```
callback(widget, content, buffers)
```

• **remove** (*bool*) – True if the callback should be unregistered.

```
on_trait_change(handler: EventHandler | None = None, name: Sentinel | str | None = None, remove: bool = False) \rightarrow None
```

DEPRECATED: Setup a handler to be called when a trait changes.

This is used to setup dynamic notifications of trait changes.

Static handlers can be created by creating methods on a HasTraits subclass with the naming convention '\_[traitname]\_changed'. Thus, to create static handler for the trait 'a', create the method \_a\_changed(self, name, old, new) (fewer arguments can be used, see below).

If remove is True and handler is not specified, all change handlers for the specified name are uninstalled.

#### **Parameters**

- handler (callable, None) A callable that is called when a trait changes. Its signature can be handler(), handler(name), handler(name, new), handler(name, old, new), or handler(name, old, new, self).
- name (list, str, None) If None, the handler will apply to all traits. If a list of str, handler will apply to all names in the list. If a str, the handler will apply just to that name.
- **remove** (bool) If False (the default), then install the handler. If True then unintall it.

# static on\_widget\_constructed(callback)

Registers a callback to be called when a widget is constructed.

The callback must have the following signature: callback(widget)

#### open()

Open a comm to the frontend if one isn't already open.

#### remove\_class(className)

Removes a class from the top level element of the widget.

Doesn't remove the class if it doesn't exist.

```
send(content, buffers=None)
```

Sends a custom msg to the widget model in the front-end.

#### **Parameters**

- **content** (*dict*) Content of the message to send.
- buffers (list of binary buffers) Binary buffers to send with message

```
send_state(key=None)
```

Sends the widget state, or a piece of it, to the front-end, if it exists.

#### Parameters

**key** (*unicode*, *or iterable* (*optional*)) – A single property's name or iterable of property names to sync with the front-end.

```
set_state(sync_data)
```

Called when a state is received from the front-end.

```
set\_trait(name: str, value: Any) \rightarrow None
```

Forcibly sets trait attribute, including read-only attributes.

```
setup_instance(**kwargs: Any) \rightarrow None
```

This is called **before** self. init is called.

### tabbable

Is widget tabbable?

#### tooltip

A tooltip caption.

```
trait_defaults(*names: str, **metadata: Any) → dict[str, Any] | Sentinel
```

Return a trait's default value or a dictionary of them

### **Notes**

Dynamically generated default values may depend on the current state of the object.

```
classmethod trait_events(name: str | None = None) → dict[str, EventHandler]
```

Get a dict of all the event handlers of this class.

### **Parameters**

**name** (*str* (*default*: *None*)) – The name of a trait of this class. If name is None then all the event handlers of this class will be returned instead.

## Return type

The event handlers associated with a trait name, or all event handlers.

```
trait_has_value(name: str) → bool
```

Returns True if the specified trait has a value.

This will return false even if getattr would return a dynamically generated default value. These default values will be recognized as existing only after they have been generated.

Example

```
class MyClass(HasTraits):
    i = Int()

mc = MyClass()
assert not mc.trait_has_value("i")
mc.i # generates a default value
assert mc.trait_has_value("i")
```

```
trait_metadata(traitname: str, key: str, default: Any = None) \rightarrow Any
```

Get metadata values for trait by key.

```
trait_names(**metadata: Any) → list[str]
```

Get a list of all the names of this class' traits.

```
trait_values(**metadata: Any) → dict[str, Any]
```

A dict of trait names and their values.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

### Return type

A dict of trait names and their values.

### **Notes**

Trait values are retrieved via getattr, any exceptions raised by traits or the operations they may trigger will result in the absence of a trait value in the result dict.

```
traits(**metadata: Any) → dict[str, TraitType[Any, Any]]
```

Get a dict of all the traits of this class. The dictionary is keyed on the name and the values are the TraitType objects.

The TraitTypes returned don't know anything about the values that the various HasTrait's instances are holding.

The metadata kwargs allow functions to be passed in which filter traits based on metadata values. The functions should take a single value as an argument and return a boolean. If any function returns False, then the trait is not included in the output. If a metadata key doesn't exist, None will be passed to the function.

```
unobserve(handler: Callable[[...], Any], names: Sentinel | str | Iterable[Sentinel | str] = traitlets.All, type: Sentinel | <math>str = 'change') \rightarrow None
```

Remove a trait change handler.

This is used to unregister handlers to trait change notifications.

### **Parameters**

- handler (callable) The callable called when a trait attribute changes.
- names (list, str, All (default: All)) The names of the traits for which the specified handler should be uninstalled. If names is All, the specified handler is uninstalled from the list of notifiers corresponding to all changes.
- **type** (*str or All (default: 'change'*)) The type of notification to filter by. If All, the specified handler is uninstalled from the list of notifiers corresponding to all types.

```
unobserve_all(name: str \mid Any = traitlets.All) \rightarrow None
```

Remove trait change handlers of any type for the specified name. If name is not specified, removes all trait notifiers.

```
widget_types = <ipywidgets.widgets.widget.WidgetRegistry object>
    widgets = {}
class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget(deck)
    __init__(deck)
    plot_binary_cb(click)
    plot_ternary_cb(click)
    calc_sweep_cb(click)
    validate_sweep_cb(click)
    get_sweep_data()
    get_deck()
    update_component_row_cb(event)
    start()
class AFL.automation.prepare.SweepBuilderWidget_Model(deck)
    __init__(deck)
    validate_sweep(tolerance, progress=None)
    calc_sweep(sweep_dict, progress)
class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget_View
    __init__()
    make_stock_grid(component_names)
    make_ternary_plot(component_names)
    make_binary_plot(component_names)
    start(component_names)
```

# AFL.automation.prepare.factory

#### **Functions**

# AFL.automation.prepare.factory.HD2OFactory

AFL.automation.prepare.factory.**HD20Factory**(name, phi\_D2O=None, sld=None, properties=None)
Create a list of H2O/D2O solutions

# AFL.automation.prepare.factory.compositionSweepFactory

AFL.automation.prepare.factory.compositionSweepFactory(name, components,  $vary\_components$ , lo, hi, num, logspace=False, properties=None, progress=None)

## AFL.automation.prepare.factory.has\_units

AFL.automation.prepare.factory.has\_units(value)

## AFL.automation.prepare.factory.is\_concentration

 $AFL. automation.prepare.factory. \textbf{is\_concentration} (\textit{value})$ 

# AFL.automation.prepare.factory.is\_mass

AFL.automation.prepare.factory.is\_mass(value)

# AFL.automation.prepare.factory.is\_molarity

AFL.automation.prepare.factory.is\_molarity(value)

## AFL.automation.prepare.factory.is volume

AFL.automation.prepare.factory.is\_volume(value)

# AFL.automation.prepare.factory.listify

AFL.automation.prepare.factory.listify(obj)

### **Classes**

*Solution*(name, components[, properties])

product

product(\*iterables, repeat=1) --> product object

## AFL.automation.prepare.factory.Solution

 $\textbf{class} \ \texttt{AFL}. \textbf{automation.prepare.factory.} \textbf{Solution} (\textit{name}, \textit{components}, \textit{properties} = None)$ 

Bases: object

 $\verb|\__init__(name, components, properties=None)|\\$ 

## **Methods**

```
__init__(name, components[, properties])
add_component_from_name(name[,
                                      properties,
contains(name)
copy([name])
from_dict(in_dict)
                                                  Create solution with identical composition at new to-
measure_out(amount[, deplete])
                                                  tal mass/volume
rename_component(old_name, new_name[, in-
place])
                                                  Setter for inline mass changes
set_mass(value)
set_properties_from_dict([properties,
                                             in-
place])
set_volume(value)
                                                  Setter for inline volume changes
to_dict()
```

## **Attributes**

concentration	
mass	Total mass of mixture.
mass_fraction	Mass fraction of components in mixture
molarity	
size	
solutes	
solvent_density	
solvent_mass	
solvent_sld	
solvent_volume	
solvents	
volume	Total volume of mixture.
volume_fraction	Volume fraction of solvents in mixture

\_\_init\_\_(name, components, properties=None)

```
__hash__()
    Needed so Solutions can be dictionary keys
to_dict()
classmethod from_dict(in_dict)
add_component_from_name(name, properties=None, inplace=False)
set_properties_from_dict(properties=None, inplace=False)
rename_component(old_name, new_name, inplace=False)
copy(name=None)
contains(name)
property size
property solutes
property solvents
__eq__(other)
     'Compare the mass, volume, and composition of two mixtures
property mass
    Total mass of mixture.
set_mass(value)
    Setter for inline mass changes
property volume
    Total volume of mixture. Only solvents are included in volume calculation
set_volume(value)
    Setter for inline volume changes
property solvent_sld
property solvent_density
property solvent_volume
property solvent_mass
property mass_fraction
    Mass fraction of components in mixture
        Returns
             • mass_fraction (dict)
             • Component mass fractions
property volume_fraction
    Volume fraction of solvents in mixture
        Returns
```

6.2. Modules 715

solvent\_fraction (dict) Component mass fractions

```
property concentration
property molarity
measure_out(amount, deplete=False)
```

Create solution with identical composition at new total mass/volume

### AFL.automation.prepare.factory.product

### class AFL.automation.prepare.factory.product

```
Bases: object
```

product(\*iterables, repeat=1) -> product object

Cartesian product of input iterables. Equivalent to nested for-loops.

For example,  $\operatorname{product}(A, B)$  returns the same as: ((x,y) for x in A for y in B). The leftmost iterators are in the outermost for-loop, so the output tuples cycle in a manner similar to an odometer (with the rightmost element changing on every iteration).

To compute the product of an iterable with itself, specify the number of repetitions with the optional repeat keyword argument. For example, product(A, repeat=4) means the same as product(A, A, A, A).

```
product(`ab', range(3)) \rightarrow (`a',0) (`a',1) (`a',2) (`b',0) (`b',1) (`b',2) product((0,1), (0,1), (0,1)) \rightarrow (0,0,0) (0,0,1) (0,1,0) (0,1,1) (1,0,0) \dots
```

```
__init__()
```

#### **Methods**

```
__init__()
```

AFL.automation.prepare.factory.**HD20Factory**(name, phi\_D2O=None, sld=None, properties=None)
Create a list of H2O/D2O solutions

AFL.automation.prepare.factory.compositionSweepFactory(name, components, vary\_components, lo, hi, num, logspace=False, properties=None, progress=None)

# AFL.automation.prepare.utilities

### **Functions**

```
make_locs(slot, nrows, ncols)
make_wellplate_locs(slot, size)
```

# AFL.automation.prepare.utilities.make\_locs

AFL.automation.prepare.utilities.make\_locs(slot, nrows, ncols)

# AFL.automation.prepare.utilities.make\_wellplate\_locs

```
AFL.automation.prepare.utilities.make_wellplate_locs(slot, size)
```

AFL.automation.prepare.utilities.make\_locs(slot, nrows, ncols)

AFL.automation.prepare.utilities.make\_wellplate\_locs(slot, size)

# **6.2.5** Sample

The Sample module provides functionality for experiment orchestration 'Sample Servers'

```
AFL.automation.sample

AFL.automation.sample_env
```

# AFL.automation.sample

# **Modules**

CastingServer

# AFL.automation.sample.CastingServer

# **Functions**

ceil(x,/)	Return the ceiling of x as an Integral.
listify(obj)	
sqrt(x,/)	Return the square root of x.

# AFL.automation.sample.CastingServer.ceil

```
AFL.automation.sample.CastingServer.ceil(x,/)
Return the ceiling of x as an Integral.
This is the smallest integer >= x.
```

# AFL.automation.sample.CastingServer.listify

```
AFL.automation.sample.CastingServer.listify(obj)
```

# AFL.automation.sample.CastingServer.sqrt

```
AFL.automation.sample.CastingServer.\mathbf{sqrt}(x,/)
Return the square root of x.
```

# Classes

CastingServer(prep_url[, overrides])	
<pre>Client([ip, port, username, interactive])</pre>	Communicate with APIServer
Driver(name[, defaults, overrides])	
<pre>OT2Client([ip, port, username, interactive])</pre>	Communicate with AFL-automation server on OT-2

# AFL.automation.sample.CastingServer.CastingServer

```
class AFL.automation.sample.CastingServer.CastingServer(prep_url, overrides=None)
    Bases: Driver
    __init__(prep_url, overrides=None)
```

# **Methods**

```
_init__(prep_url[, overrides])
assign_targets(protocols, target_map)
bulk_cast_films(**spec)
                                                   Spec should contain sample_name and a protocol
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
execute(**kwargs)
                                                   Gather all inherited static class-level dictionaries
gather_defaults()
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
init_casting_manifest([attrs, overwrite])
log_event(manifest_key, event[, write])
post_execute(**kwargs)
                                                   Executed after each call to execute
pre_execute(**kwargs)
                                                   Executed before each call to execute
prepare_and_cast(**sample)
prepare_casting_stocks(**spec)
                                                   Spec should contain sample_name and a protocol
queued()
quickbar()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
                                                   Set data in the DataPacket object
set_data(data)
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
unqueued()
update_status(value)
```

# **Attributes**

```
defaults
```

```
defaults = {'manifest': '/home/af1642/', 'manifest_cast': '/home/af1642/',
'manifest_prep': '/home/af1642/'}
__init__(prep_url, overrides=None)
init_casting_manifest(attrs=None, overwrite=False)
status()
update_status(value)
log_event(manifest_key, event, write=True)
assign_targets(protocols, target_map)
prepare_casting_stocks(**spec)
    Spec should contain sample_name and a protocol
bulk_cast_films(**spec)
    Spec should contain sample name and a protocol
prepare_and_cast(**sample)
deposit_obj(obj, uid=None)
    Store an object in the dropbox
        Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
    Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
```

Executed after each call to execute

All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden by subclasses.

```
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
set_config(**kwargs)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
             • data (dict) – Dictionary of data to store in the driver object
             • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
set_object(serialized=True, **kw)
set_sample(sample_name, sample_uuid=None, **kwargs)
unqueued()
```

# AFL.automation.sample.CastingServer.Client

Bases: object

Communicate with APIServer

This class provides an interface to generate HTTP REST requests that are sent to an APIServer, monitor the status of those requests, and retrieve the results of those requests. It is intended to be used as a client to the APIServer class.

```
__init__(ip=None, port='5000', username=None, interactive=False)
```

# **Methods**

```
_init__([ip, port, username, interactive])
clear_history()
clear_queue()
debug(state)
deposit_obj(obj[, uid])
                                                  Deposit an object in the dropbox obj: object, the ob-
                                                  ject to deposit id: str, the uuid to deposit the object
                                                  under if not specified, a new uuid will be generated
driver_status()
enqueue([interactive])
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
get_unqueued_commands([inherit_commands])
halt()
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
```

continues on next page

Table 58 - continued from previous page

```
remove_item(uuid)
 reset_queue_daemon()
 retrieve_obj(uid[, delete])
                                                  Retrieve an object from the dropbox id: str, the uuid
                                                  of the object to retrieve delete: bool, if True, delete
                                                  the object after retrieving
 server_cmd(cmd, **kwargs)
 set_config([interactive])
 set_driver_object(**kw)
 set_object([serialize])
 unqueued_base(**kwargs)
 wait([target_uuid, interval, for_history, ...])
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
query_driver(**kwargs)
```

```
reset_queue_daemon()
     pause(state)
     clear_queue()
     clear_history()
     debug(state)
     halt()
     queue_state()
     remove_item(uuid)
     move_item(uuid, pos)
     set_driver_object(**kw)
     get_driver_object(name)
     deposit_obj(obj, uid=None)
          Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
          under if not specified, a new uuid will be generated
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
          the object after retrieving
     set_object(serialize=True, **kw)
     get_object(name, serialize=True)
AFL.automation.sample.CastingServer.Driver
class AFL.automation.sample.CastingServer.Driver(name, defaults=None, overrides=None)
     Bases: object
```

```
__init__(name, defaults=None, overrides=None)
```

# **Methods**

```
_init__(name[, defaults, overrides])
                                                    Store an object in the dropbox
 deposit_obj(obj[, uid])
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
```

• uid(str) – The uuid to store the object under

# AFL.automation.sample.CastingServer.OT2Client

Bases: Client

Communicate with AFL-automation server on OT-2

This class maps pipettor functions to HTTP REST requests that are sent to the AFL-automation server \_\_init\_\_(ip=None, port='5000', username=None, interactive=False)

# **Methods**

```
__init__([ip, port, username, interactive])
aspirate_rate(rate)
clear_history()
clear_queue()
debug(state)
deposit_obj(obj[, uid])
                                                   Deposit an object in the dropbox obj: object, the ob-
                                                   ject to deposit id: str, the uuid to deposit the object
                                                   under if not specified, a new uuid will be generated
dispense_rate(rate)
driver_status()
enqueue([interactive])
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
```

continues on next page

Table 59 - continued from previous page

```
get_unqueued_commmands([inherit_commands])
halt()
home()
load_instrument(name, mount, tip_rack_slots)
load_labware(name, slot)
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
queue_state()
remove_item(uuid)
reset_queue_daemon()
reset_tipracks([mount])
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox id: str, the uuid
                                                   of the object to retrieve delete: bool, if True, delete
                                                   the object after retrieving
server_cmd(cmd, **kwargs)
set_config([interactive])
set_driver_object(**kw)
set_object([serialize])
                                                   Transfer fluid from one location to another
transfer(source, dest, volume[, interactive])
unqueued_base(**kwargs)
wait([target_uuid, interval, for_history, ...])
```

transfer(source, dest, volume, interactive=None, \*\*kwargs)

Transfer fluid from one location to another

#### **Parameters**

• source (str or list of str Source wells to transfer from. Wells should be specified as three) — character strings with the first character being the slot number.

```
    dest(str or list of str) – Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
    volume (float) – volume of fluid to transfer in microliters
```

```
reset_tipracks(mount='both')
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
aspirate_rate(rate)
dispense_rate(rate)
home()
__init__(ip=None, port='5000', username=None, interactive=False)
clear_history()
clear_queue()
debug(state)
deposit_obj(obj, uid=None)
    Deposit an object in the dropbox obj: object, the object to deposit id: str, the uuid to deposit the object
    under if not specified, a new uuid will be generated
driver_status()
enqueue(interactive=None, **kwargs)
enqueued_base(**kwargs)
classmethod from_server_name(server_name, **kwargs)
get_config(name, print_console=True, interactive=None)
get_driver_object(name)
get_object(name, serialize=True)
get_queue()
get_queued_commands(inherit_commands=True)
get_quickbar()
get_server_time()
get_unqueued_commands(inherit_commands=True)
halt()
logged_in()
login(username, populate_commands=True)
move_item(uuid, pos)
```

```
pause(state)
     query_driver(**kwargs)
     queue_state()
     remove_item(uuid)
     reset_queue_daemon()
     retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox id: str, the uuid of the object to retrieve delete: bool, if True, delete
          the object after retrieving
     server_cmd(cmd, **kwargs)
     set_config(interactive=None, **kwargs)
     set_driver_object(**kw)
     set_object(serialize=True, **kw)
     unqueued_base(**kwargs)
     wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
class AFL.automation.sample.CastingServer.CastingServer(prep_url, overrides=None)
     defaults = {'manifest': '/home/af1642/', 'manifest_cast': '/home/af1642/',
     'manifest_prep': '/home/afl642/'}
     __init__(prep_url, overrides=None)
     init_casting_manifest(attrs=None, overwrite=False)
     status()
     update_status(value)
     log_event(manifest_key, event, write=True)
     assign_targets(protocols, target_map)
     prepare_casting_stocks(**spec)
          Spec should contain sample_name and a protocol
     bulk_cast_films(**spec)
          Spec should contain sample_name and a protocol
     prepare_and_cast(**sample)
```

# AFL.automation.sample\_env

# Modules

*TemperatureDeck* 

# AFL.automation.sample\_env.TemperatureDeck

# **Classes**

Driver(name[, defaults, overrides])

TemperatureDeck([overrides])

# ${\bf AFL}. automation. sample\_env. Temperature {\bf Deck. Driver}$

 $\textbf{class} \hspace{0.1cm} \textbf{AFL.automation.sample\_env.} Temperature \textbf{Deck.} \\ \textbf{Driver} (\textit{name}, \textit{defaults} = \textit{None}, \textit{overrides} = \textit{None}) \\$ 

Bases: object

\_\_init\_\_(name, defaults=None, overrides=None)

# **Methods**

```
_init__(name[, defaults, overrides])
 deposit_obj(obj[, uid])
                                                    Store an object in the dropbox
 execute(**kwargs)
 gather_defaults()
                                                    Gather all inherited static class-level dictionaries
                                                    called default.
 get_config(name[, print_console])
 get_configs([print_console])
 get_object(name[, serialize])
 get_sample()
                                                    Executed after each call to execute
 post_execute(**kwargs)
 pre_execute(**kwargs)
                                                    Executed before each call to execute
 queued()
 quickbar()
 reset_sample()
 retrieve_obj(uid[, delete])
                                                    Retrieve an object from the dropbox
 set_config(**kwargs)
 set_data(data)
                                                    Set data in the DataPacket object
 set_object([serialized])
 set_sample(sample_name[, sample_uuid])
 status()
 unqueued()
unqueued()
queued()
quickbar()
__init__(name, defaults=None, overrides=None)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
set_config(**kwargs)
get_config(name, print_console=False)
```

```
get_configs(print_console=False)
set_sample(sample_name, sample_uuid=None, **kwargs)
get_sample()
reset_sample()
status()
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
execute(**kwargs)
set_object(serialized=True, **kw)
get_object(name, serialize=True)
set_data(data: dict)
     Set data in the DataPacket object
         Parameters
              • data (dict) – Dictionary of data to store in the driver object
              • variables (Note! if the keys in data are not system or sample)
     :param : :param they will be erased at the end of this function call.:
retrieve_obj(uid, delete=True)
     Retrieve an object from the dropbox
         Parameters
             uid (str) – The uuid of the file to retrieve
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
              • obj (object) – The object to store in the dropbox
              • uid (str) – The uuid to store the object under
```

# AFL.automation.sample\_env.TemperatureDeck.TemperatureDeck

```
class AFL.automation.sample_env.TemperatureDeck.TemperatureDeck(overrides=None)
    Bases: Driver
    __init__(overrides=None)
```

#### **Methods**

```
__init__([overrides])
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
execute(**kwargs)
gather_defaults()
                                                   Gather all inherited static class-level dictionaries
                                                   called default.
get_config(name[, print_console])
get_configs([print_console])
get_object(name[, serialize])
get_sample()
move_temp(temperature[, wait, tolerance])
post_execute(**kwargs)
                                                   Executed after each call to execute
pre_execute(**kwargs)
                                                   Executed before each call to execute
queued()
quickbar()
read_temp()
reset_sample()
retrieve_obj(uid[, delete])
                                                   Retrieve an object from the dropbox
set_config(**kwargs)
set_data(data)
                                                   Set data in the DataPacket object
set_object([serialized])
set_sample(sample_name[, sample_uuid])
set_temp(sp)
status()
unqueued()
```

# **Attributes**

```
defaults
```

```
defaults = {'serial_port': '/dev/ttyACMO', 'temperature_move_sleep': 0.2,
'temperature_move_timeout':
__init__(overrides=None)
set_temp(sp)
move_temp(temperature, wait=30, tolerance=0.1)
status()
read_temp()
deposit_obj(obj, uid=None)
     Store an object in the dropbox
         Parameters
             • obj (object) – The object to store in the dropbox
             • uid (str) – The uuid to store the object under
execute(**kwargs)
classmethod gather_defaults()
     Gather all inherited static class-level dictionaries called default.
get_config(name, print_console=False)
get_configs(print_console=False)
get_object(name, serialize=True)
get_sample()
post_execute(**kwargs)
     Executed after each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
pre_execute(**kwargs)
     Executed before each call to execute
     All of the kwargs passed to execute are also pass to this method. It is expected that this method be overridden
     by subclasses.
queued()
quickbar()
reset_sample()
```

```
retrieve_obj(uid, delete=True)
          Retrieve an object from the dropbox
              Parameters
                 uid (str) – The uuid of the file to retrieve
     set_config(**kwargs)
     set_data(data: dict)
          Set data in the DataPacket object
              Parameters
                  • data (dict) – Dictionary of data to store in the driver object
                  • variables (Note! if the keys in data are not system or sample)
          :param : :param they will be erased at the end of this function call.:
     set_object(serialized=True, **kw)
     set_sample(sample_name, sample_uuid=None, **kwargs)
     unqueued()
class AFL.automation.sample_env.TemperatureDeck.TemperatureDeck(overrides=None)
     defaults = {'serial_port': '/dev/ttyACMO', 'temperature_move_sleep': 0.2,
     'temperature_move_timeout': 900}
     __init__(overrides=None)
     set_temp(sp)
     move_temp(temperature, wait=30, tolerance=0.1)
     status()
     read_temp()
```

# 6.2.6 Shared

The Shared module provides common utilities and functionality used across the AFL-automation framework.

AFL.automation.shared

# AFL.automation.shared

# **Modules**

DataLabelerWidget

DatasetWidget

DiffractionLabeler

MutableQueue

PersistentConfig

ServerDiscovery

exceptions

serialization

units

utilities

widgetui

# AFL.automation.shared.DataLabelerWidget

# **Functions**

sqrt(x, t) Return the square root of x.	sqrt(x,/)	Return the square root of x.
---	-----------	------------------------------

# AFL.automation.shared.DataLabelerWidget.sqrt

AFL.automation.shared.DataLabelerWidget.sqrt(x,/)Return the square root of x.

# **Classes**

```
DataLabelerModel(dataset)

DataLabelerView()

DataLabelerWidget(input_dataset, ...[, ...])

OrdinalEncoder(*[, categories, dtype, ...])

Encode categorical features as an integer array.
```

# AFL.automation.shared.DataLabelerWidget.DataLabelerModel

```
class AFL.automation.shared.DataLabelerWidget.DataLabelerModel(dataset: Dataset)
    Bases: object
    __init__(dataset: Dataset)
```

#### **Methods**

```
__init__(dataset)

get_peaks(model[, qstar, max_order])

init_models()

ordinal_phase_labels()

__init__(dataset: Dataset)

ordinal_phase_labels()

init_models()

get_peaks(model, qstar=None, max_order=4)
```

# AFL.automation.shared.DataLabelerWidget.DataLabelerView

# **Methods**

```
__init__()

add_vertical_line(x[, y0, y1, row, col, line_kw])

remove_vertical_lines()

run(x, y, all_compositions, composition, ...)

update_composition_colors(colors)

update_plot(x, y, composition)
```

```
__init__()
update_plot(x, y, composition)
remove_vertical_lines()
add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
update_composition_colors(colors)
run(x, y, all_compositions, composition, models, ternary, components)
```

# AFL.automation.shared.DataLabelerWidget.DataLabelerWidget

Bases: object

\_\_init\_\_(input\_dataset: Dataset, sas\_variable: str, composition\_variable: str | List[str], sample\_dim: str = 'sample', fit\_variable: str | None = None)

#### **Parameters**

- dataset (xr.Dataset) Dataset from AFL
- **sas\_variable** (*str*) Name of data variable in the *xarray.Dataset* that holds the scattering data
- **composition\_variable** (*str | List[str]*) Name of data variable in the *xar-ray.Dataset* that holds the composition. If the composition is split across multiple variables, pass in a list of variables.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to each sample or measurement. This is typically named 'sample' in much of the AFL agent codebase.
- **fit\_variable** (*Optional[str]*) If not none, this data will be plotted along with the sas\_variable data. This data variable should have the same shape as sas\_variable.

# **Methods**

```
__init__(input_dataset, sas_variable, ...[, ...])

change_model_callback(data)

change_norder_callback(figure, location, click)

composition_click_callback(figure, location, ...)

draw_peaks(peaks)

goto_callback(click)

label(label)

next_button_callback(click)

prev_button_callback(click)

run()

update_plot()
```

\_\_init\_\_(input\_dataset: Dataset, sas\_variable: str, composition\_variable: str | List[str], sample\_dim: str = 'sample', fit\_variable: str | None = None')

### **Parameters**

- dataset (xr.Dataset) Dataset from AFL
- **sas\_variable** (*str*) Name of data variable in the *xarray.Dataset* that holds the scattering data
- **composition\_variable** (*str | List[str]*) Name of data variable in the *xar-ray.Dataset* that holds the composition. If the composition is split across multiple variables, pass in a list of variables.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to each sample or measurement. This is typically named 'sample' in much of the AFL agent codebase.
- **fit\_variable** (*Optional[str]*) If not none, this data will be plotted along with the sas\_variable data. This data variable should have the same shape as sas\_variable.

```
next_button_callback(click)
prev_button_callback(click)
goto_callback(click)
composition_click_callback(figure, location, click)
update_plot()
```

max\_categories=None)

```
draw_peaks(peaks)
change_qstar_callback(figure, location, click)
change_model_callback(data)
change_norder_callback(data)
label(label)
run()
```

# AFL.automation.shared.DataLabelerWidget.OrdinalEncoder

Bases: OneToOneFeatureMixin, \_BaseEncoder

Encode categorical features as an integer array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are converted to ordinal integers. This results in a single column of integers (0 to n\_categories - 1) per feature.

Read more in the User Guide. For a comparison of different encoders, refer to: sphx\_glr\_auto\_examples\_preprocessing\_plot\_target\_encoder.py.

Added in version 0.20.

#### **Parameters**

- categories ('auto' or a list of array-like, default='auto') Categories (unique values) per feature:
  - 'auto': Determine categories automatically from the training data.
  - list: categories[i] holds the categories expected in the ith column. The passed categories should not mix strings and numeric values, and should be sorted in case of numeric values.

The used categories can be found in the categories\_ attribute.

- **dtype** (*number type*, *default=np.float64*) Desired dtype of output.
- handle\_unknown ({'error', 'use\_encoded\_value'}, default='error') When set to 'error' an error will be raised in case an unknown categorical feature is present during transform. When set to 'use\_encoded\_value', the encoded value of unknown categories will be set to the value given for the parameter unknown\_value. In inverse\_transform(), an unknown category will be denoted as None.

Added in version 0.24.

• unknown\_value (int or np.nan, default=None) — When the parameter handle\_unknown is set to 'use\_encoded\_value', this parameter is required and will set the

encoded value of unknown categories. It has to be distinct from the values used to encode any of the categories in *fit*. If set to np.nan, the *dtype* parameter must be a float dtype.

Added in version 0.24.

• **encoded\_missing\_value** (*int or np.nan*, *default=np.nan*) – Encoded value of missing categories. If set to *np.nan*, then the *dtype* parameter must be a float dtype.

Added in version 1.1.

- min\_frequency (int or float, default=None) Specifies the minimum frequency below which a category will be considered infrequent.
  - If int, categories with a smaller cardinality will be considered infrequent.
  - If float, categories with a smaller cardinality than min\_frequency \* n\_samples will be considered infrequent.

Added in version 1.3: Read more in the User Guide.

• max\_categories (int, default=None) – Specifies an upper limit to the number of output categories for each input feature when considering infrequent categories. If there are infrequent categories, max\_categories includes the category representing the infrequent categories along with the frequent categories. If None, there is no limit to the number of output features.

 $max\_categories$  do **not** take into account missing or unknown categories. Setting  $unknown\_value$  or  $encoded\_missing\_value$  to an integer will increase the number of unique integer codes by one each. This can result in up to  $max\_categories + 2$  integer codes.

Added in version 1.3: Read more in the User Guide.

### categories\_

The categories of each feature determined during fit (in order of the features in X and corresponding with the output of transform). This does not include categories that weren't seen during fit.

#### **Type**

list of arrays

### n\_features\_in\_

Number of features seen during fit.

Added in version 1.0.

### **Type**

int

# feature\_names\_in\_

Names of features seen during fit. Defined only when *X* has feature names that are all strings.

Added in version 1.0.

#### **Type**

ndarray of shape (n features in ,)

#### infrequent\_categories\_

Defined only if infrequent categories are enabled by setting *min\_frequency* or *max\_categories* to a non-default value. *infrequent\_categories\_[i]* are the infrequent categories for feature *i*. If the feature *i* has no infrequent categories *infrequent\_categories\_[i]* is None.

Added in version 1.3.

# **Type**

list of ndarray

#### See also

### OneHotEncoder

Performs a one-hot encoding of categorical features. This encoding is suitable for low to medium cardinality categorical variables, both in supervised and unsupervised settings.

### TargetEncoder

Encodes categorical features using supervised signal in a classification or regression pipeline. This encoding is typically suitable for high cardinality categorical variables.

#### LabelEncoder

Encodes target labels with values between 0 and n\_classes-1.

#### **Notes**

With a high proportion of *nan* values, inferring categories becomes slow with Python versions before 3.10. The handling of *nan* values was improved from Python 3.10 onwards, (c.f. bpo-43475).

# **Examples**

Given a dataset with two features, we let the encoder find the unique values per feature and transform the data to an ordinal encoding.

By default, OrdinalEncoder is lenient towards missing values by propagating them.

You can use the parameter encoded\_missing\_value to encode missing values.

(continued from previous page)

```
[ 0., 1.],
[ 0., -1.]])
```

Infrequent categories are enabled by setting *max\_categories* or *min\_frequency*. In the following example, "a" and "d" are considered infrequent and grouped together into a single category, "b" and "c" are their own categories, unknown values are encoded as 3 and missing values are encoded as 4.

```
>>> X_train = np.array(
        [["a"] * 5 + ["b"] * 20 + ["c"] * 10 + ["d"] * 3 + [np.nan]],
. . .
        dtype=object).T
>>> enc = OrdinalEncoder(
        handle_unknown="use_encoded_value", unknown_value=3,
        max_categories=3, encoded_missing_value=4)
>>> _ = enc.fit(X_train)
>>> X_test = np.array([["a"], ["b"], ["c"], ["d"], ["e"], [np.nan]], dtype=object)
>>> enc.transform(X_test)
array([[2.],
       [0.],
       [1.],
       [2.],
       [3.],
       [4.]])
```

\_\_init\_\_(\*, categories='auto', dtype=<class 'numpy.float64'>, handle\_unknown='error', unknown\_value=None, encoded\_missing\_value=nan, min\_frequency=None, max\_categories=None)

### **Methods**

init(*[, categories, dtype,])	
fit(X[,y])	Fit the OrdinalEncoder to X.
$fit_transform(X[,y])$	Fit to data, then transform it.
<pre>get_feature_names_out([input_features])</pre>	Get output feature names for transformation.
<pre>get_metadata_routing()</pre>	Get metadata routing of this object.
<pre>get_params([deep])</pre>	Get parameters for this estimator.
<pre>inverse_transform(X)</pre>	Convert the data back to the original representation.
<pre>set_output(*[, transform])</pre>	Set output container.
<pre>set_params(**params)</pre>	Set the parameters of this estimator.
transform(X)	Transform X to ordinal codes.

# **Attributes**

infrequent\_categories\_

Infrequent categories for each feature.

```
__init__(*, categories='auto', dtype=<class 'numpy.float64'>, handle_unknown='error', unknown_value=None, encoded_missing_value=nan, min_frequency=None, max_categories=None)
```

```
fit(X, y=None)
```

Fit the OrdinalEncoder to X.

#### **Parameters**

- **X** (array-like of shape (n\_samples, n\_features)) The data to determine the categories of each feature.
- **y** (*None*) Ignored. This parameter exists only for compatibility with Pipeline.

#### Returns

self - Fitted encoder.

# Return type

object

#### transform(X)

Transform X to ordinal codes.

#### **Parameters**

 $\mathbf{X}$  (array-like of shape (n\_samples, n\_features)) – The data to encode.

#### Returns

**X\_out** – Transformed input.

### **Return type**

ndarray of shape (n\_samples, n\_features)

# $inverse\_transform(X)$

Convert the data back to the original representation.

#### **Parameters**

 $\boldsymbol{\mathtt{X}}$  (array-like of shape (n\_samples, n\_encoded\_features)) – The transformed data.

### Returns

 $X_{tr}$  – Inverse transformed array.

# Return type

ndarray of shape (n\_samples, n\_features)

# **fit\_transform**(*X*, *y=None*, \*\*fit\_params)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit\_params* and returns a transformed version of *X*.

#### **Parameters**

- X (array-like of shape (n\_samples, n\_features)) Input samples.
- y (array-like of shape (n\_samples,) or (n\_samples, n\_outputs), default=None) Target values (None for unsupervised transformations).

• **\*\*fit\_params** (*dict*) – Additional fit parameters.

#### Returns

**X\_new** – Transformed array.

# **Return type**

ndarray array of shape (n\_samples, n\_features\_new)

# get\_feature\_names\_out(input\_features=None)

Get output feature names for transformation.

#### **Parameters**

input\_features (array-like of str or None, default=None) - Input features.

- If *input\_features* is *None*, then *feature\_names\_in\_* is used as feature names in. If *feature\_names\_in\_* is not defined, then the following input feature names are generated: ["x0", "x1", ..., "x(n\_features\_in\_ 1)"].
- If *input\_features* is an array-like, then *input\_features* must match *feature\_names\_in\_* if *feature\_names\_in\_* is defined.

#### Returns

**feature\_names\_out** – Same as input features.

### Return type

ndarray of str objects

# get\_metadata\_routing()

Get metadata routing of this object.

Please check User Guide on how the routing mechanism works.

# Returns

**routing** – A MetadataRequest encapsulating routing information.

# Return type

MetadataRequest

#### get\_params(deep=True)

Get parameters for this estimator.

### **Parameters**

**deep** (*bool*, *default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

#### Returns

params – Parameter names mapped to their values.

# **Return type**

dict

# property infrequent\_categories\_

Infrequent categories for each feature.

```
set_output(*, transform=None)
```

Set output container.

See sphx\_glr\_auto\_examples\_miscellaneous\_plot\_set\_output.py for an example on how to use the API.

# **Parameters**

**transform**({"default", "pandas", "polars"}, default=None)—Configure output of *transform* and *fit\_transform*.

- "default": Default output format of a transformer
- "pandas": DataFrame output
- "polars": Polars output
- None: Transform configuration is unchanged

Added in version 1.4: "polars" option was added.

#### Returns

**self** – Estimator instance.

# **Return type**

estimator instance

# set\_params(\*\*params)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as Pipeline). The latter have parameters of the form <component>\_\_<parameter> so that it's possible to update each component of a nested object.

#### **Parameters**

\*\*params (dict) – Estimator parameters.

#### Returns

**self** – Estimator instance.

# Return type

estimator instance

class AFL.automation.shared.DataLabelerWidget.DataLabelerWidget(input\_dataset: Dataset,

```
sas_variable: str,
composition_variable: str |
List[str], sample_dim: str =
'sample', fit_variable: str | None
= None)
```

\_\_init\_\_(input\_dataset: Dataset, sas\_variable: str, composition\_variable: str | List[str], sample\_dim: str = 'sample', fit\_variable: str | None = None)

#### **Parameters**

- dataset (xr.Dataset) Dataset from AFL
- **sas\_variable** (*str*) Name of data variable in the *xarray.Dataset* that holds the scattering data
- **composition\_variable** (*str | List[str]*) Name of data variable in the *xar-ray.Dataset* that holds the composition. If the composition is split across multiple variables, pass in a list of variables.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to each sample or measurement. This is typically named 'sample' in much of the AFL agent codebase.
- **fit\_variable** (*Optional[str]*) If not none, this data will be plotted along with the sas variable data. This data variable should have the same shape as sas variable.

next\_button\_callback(click)

prev\_button\_callback(click)

```
goto_callback(click)
     composition_click_callback(figure, location, click)
     update_plot()
     draw_peaks(peaks)
     change_qstar_callback(figure, location, click)
     change_model_callback(data)
     change_norder_callback(data)
     label(label)
     run()
class AFL.automation.shared.DataLabelerWidget.DataLabelerModel(dataset: Dataset)
     __init__(dataset: Dataset)
     ordinal_phase_labels()
     init_models()
     get_peaks(model, qstar=None, max_order=4)
class AFL.automation.shared.DataLabelerWidget.DataLabelerView
     __init__()
     update_plot(x, y, composition)
     remove_vertical_lines()
     add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
     update_composition_colors(colors)
     run(x, y, all\_compositions, composition, models, ternary, components)
```

# AFL.automation.shared.DatasetWidget

# **Classes**

748

# AFL.automation.shared.DatasetWidget.DatasetWidget

```
class AFL.automation.shared.DatasetWidget.DatasetWidget(dataset: Dataset, sample_dim: str = sample', scatt\_variables: List[str] \mid None = None, comps\_variable: str \mid None = None, comps\_color\_variable: str \mid None = None, state = none, sta
```

Bases: object

```
__init__(dataset: Dataset, sample_dim: str = 'sample', scatt_variables: List[str] | None = None, comps_variable: str | None = None, comps_color_variable: str | None = None, xmin: float = 0.001, xmax: float = 1.0)
```

Interactive widget for viewing compositionally varying scattering data

#### **Parameters**

- **dataset** (*xr.Dataset*) *xarray.Dataset* containing scattering data and compositions to be plotted.
- **sample\_dim** (*str*) The name of the *xarray* dimension corresponding to sample variation, typically "sample"
- **comps\_variable** (*Optional[str]*) The name of the *xarray* variable to plot as compositional data. Optional, if not specified, can be customized in the GUI.

Only the first two columns of the data will be used in the plot. If the compositions are in separate `xarray.DataArray`s, they should be grouped into single `xarray.DataArray`s like so:

```
`python ds['comps'] = ds[['A','B','C']].to_array('component').
transpose(...,'component') `
```

- **comps\_color\_variable** (*Optional[str]*) The name of the *xarray* variable to use as the colorscale of the compositional data scatter plot. Optional, if not specified, can be customized in the GUI.
- xmin (float) Set the default q-range of the scattering data. Can be customized in the GUI
- xmax (float) Set the default q-range of the scattering data. Can be customized in the GUI
- Usage
- ----
- ```python -
- DatasetWidget(ds) (widget =)
- widget.run()
- ``` \_

# **Methods**

```
_init__(dataset[, sample_dim, ...])
                                                Interactive widget for viewing compositionally vary-
                                                ing scattering data
apply_isel(*args)
apply_sel(*args)
combine_vars(*args)
composition_click_callback(figure, location,
extract_var(*args)
get_comps()
goto_callback(*args)
initialize_plots(*args)
next_button_callback(*args)
prev_button_callback(*args)
reset_dataset(*args)
run()
update_colors(*args)
update_composition_plot()
update_dropdowns(*args)
update_extract_coords(change)
update_plots()
update_sample_dim(*args)
update_scattering_plot()
```

```
__init__(dataset: Dataset, sample_dim: str = 'sample', scatt_variables: List[str] | None = None, comps_variable: str | None = None, comps_color_variable: str | None = None, xmin: float = 0.001, xmax: float = 1.0)
```

Interactive widget for viewing compositionally varying scattering data

#### **Parameters**

• **dataset** (xr.Dataset) – xarray.Dataset containing scattering data and compositions to be plotted.

- **sample\_dim**(*str*) The name of the *xarray* dimension corresponding to sample variation, typically "sample"
- **comps\_variable** (*Optional[str]*) The name of the *xarray* variable to plot as compositional data. Optional, if not specified, can be customized in the GUI.

Only the first two columns of the data will be used in the plot. If the compositions are in separate `xarray.DataArray`s, they should be grouped into single `xarray.DataArray`s like so:

```
`python ds['comps'] = ds[['A','B','C']].to_array('component').
transpose(...,'component') `
```

- **comps\_color\_variable** (*Optional[str]*) The name of the *xarray* variable to use as the colorscale of the compositional data scatter plot. Optional, if not specified, can be customized in the GUI.
- **xmin** (*float*) Set the default q-range of the scattering data. Can be customized in the GUI
- xmax (float) Set the default q-range of the scattering data. Can be customized in the GUI

```
Usage
-----
```python -
DatasetWidget(ds) (widget =)
widget.run()
```
```

```
next_button_callback(*args)
prev_button_callback(*args)
goto_callback(*args)
composition_click_callback(figure, location, click)
update_composition_plot()
update_scattering_plot()
update_plots()
get_comps()
initialize_plots(*args)
update_colors(*args)
apply_sel(*args)
apply_isel(*args)
extract_var(*args)
```

combine\_vars(\*args)

```
reset_dataset(*args)
update_dropdowns(*args)
update_sample_dim(*args)
update_extract_coords(change)
run()
```

# AFL.automation.shared.DatasetWidget.DatasetWidget\_Model

# **Methods**

```
__init__(dataset, sample_dim)

apply_isel(kw)

apply_sel(kw)

combine_vars(combined_var, to_combine_vars)

extract_var(extract_from_var, extract_from_coord)

get_composition(variable)

get_non_sample_dims(var)

get_scattering(variable, index)

reset_dataset()

split_vars() Heuristically try to split vars into categories
```

#### **Attributes**

```
__init__(dataset: Dataset, sample_dim: str)
property dataset
reset_dataset()
```

```
split_vars()
    Heuristically try to split vars into categories
get_non_sample_dims(var: str)
apply_sel(kw)
apply_isel(kw)
combine_vars(combined_var: str, to_combine_vars: List[str])
extract_var(extract_from_var: str, extract_from_coord: str)
get_composition(variable)
get_scattering(variable, index)
```

## AFL.automation.shared.DatasetWidget.DatasetWidget\_View

```
 \textbf{class AFL}. \textbf{automation.shared.DatasetWidget.DatasetWidget_View} (initial\_scatt\_variables: List[str] \mid \\ None = None, \\ initial\_comps\_variable: str \mid None = \\ None, initial\_comps\_color\_variable: \\ str \mid None = None)  Bases: object
```

\_\_init\_\_(initial\_scatt\_variables: List[str] | None = None, initial\_comps\_variable: str | None = None, initial\_comps\_color\_variable: str | None = None)

```
_init__([initial_scatt_variables, ...])
 init_buttons()
 init_checkboxes()
 init_dropdowns()
 init_inputs()
 init_plots()
 plot_comp(x, y[, z, xname, yname, zname, colors])
 plot_sas(x, y[, name, append])
 run()
 update_colorscale([colors])
 update_dropdowns([sample_vars, scatt_vars, ...])
 update_selected(**kw)
__init__(initial_scatt_variables: List[str] | None = None, initial_comps_variable: str | None = None,
          initial_comps_color_variable: str | None = None)
update_colorscale(colors=None)
update_selected(**kw)
update_dropdowns(sample_vars=None, scatt_vars=None, comp_vars=None)
plot_sas(x, y, name='SAS', append=False)
plot\_comp(x, y, z=None, xname='x', yname='y', zname='z', colors=None)
init_plots()
init_buttons()
init_checkboxes()
init_dropdowns()
init_inputs()
run()
```

## AFL.automation.shared.DatasetWidget.defaultdict

## class AFL.automation.shared.DatasetWidget.defaultdict

Bases: dict

 $defaultdict(default\_factory=None, /, [...]) \rightarrow dict with default factory$ 

The default factory is called without arguments to produce a new value when a key is not present, in \_\_getitem\_\_ only. A defaultdict compares equal to a dict with the same items. All remaining arguments are treated the same as if they were passed to the dict constructor, including keyword arguments.

### **Methods**

| init(*args, **kwargs)          |  |
|--------------------------------|--|
| clear()                        |  |
| copy()                         |  |
|                                |  |
| <pre>fromkeys([value])</pre>   | Create a new dictionary with keys from iterable and values set to value.   |
| <pre>get(key[, default])</pre> | Return the value for key if key is in the dictionary, else default.  |
| <pre>items()</pre>             |  |
| keys()                         |  |
| <i>pop</i> (k[,d])             | If the key is not found, return the default if given; otherwise, raise a KeyError.   |
| <pre>popitem()</pre>           | Remove and return a (key, value) pair as a 2-tuple.  |
| setdefault(key[, default])     | Insert key with a value of default if key is not in the dictionary.  |
| update([E, ]**F)               | If E is present and has a .keys() method, then does: for k in E: $D[k] = E[k]$ If E is present and lacks a .keys() method, then does: for k, v in E: $D[k] = v$ In either case, this is followed by: for k in F: $D[k] = F[k]$ |
| values()                       |  |

## **Attributes**

| default_factory                                       | Factory for default value called bymissing(). |
|---|---|
| init(*args, **kwargs)                                 |   |
| $clear() \rightarrow None$ . Remove all items from D. |   |
| <b>copy</b> () $\rightarrow$ a shallow copy of D.     |   |

# default\_factory Factory for default value called by \_\_missing\_\_(). fromkeys(value=None,/) Create a new dictionary with keys from iterable and values set to value. get(key, default=None, /) Return the value for key if key is in the dictionary, else default. **items**() $\rightarrow$ a set-like object providing a view on D's items **keys()** $\rightarrow$ a set-like object providing a view on D's keys $pop(k[,d]) \rightarrow v$ , remove specified key and return the corresponding value. If the key is not found, return the default if given; otherwise, raise a KeyError. popitem() Remove and return a (key, value) pair as a 2-tuple. Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty. setdefault(key, default=None, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

```
update([E, ]^{**F}) \rightarrow None. Update D from dict/iterable E and F.
```

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

**values()**  $\rightarrow$  an object providing a view on D's values

```
class AFL.automation.shared.DatasetWidget.DatasetWidget(dataset: Dataset, sample_dim: str =
                                                                    'sample', scatt_variables: List[str] | None =
                                                                   None, comps\_variable: str | None = None,
                                                                   comps_color_variable: str | None = None,
                                                                   xmin: float = 0.001, xmax: float = 1.0)
```

```
__init__(dataset: Dataset, sample_dim: str = 'sample', scatt_variables: List[str] | None = None,
           comps\_variable: str \mid None = None, comps\_color\_variable: str \mid None = None, xmin: float =
           0.001, xmax: float = 1.0)
```

Interactive widget for viewing compositionally varying scattering data

## **Parameters**

- dataset (xr.Dataset) xarray.Dataset containing scattering data and compositions to be plotted.
- **sample\_dim** (str) The name of the xarray dimension corresponding to sample variation, typically "sample"
- comps\_variable (Optional[str]) The name of the xarray variable to plot as compositional data. Optional, if not specified, can be customized in the GUI.

Only the first two columns of the data will be used in the plot. If the compositions are in separate `xarray.DataArray`s, they should be grouped into single `xarray.DataArray`s like so:

```
`python ds['comps'] = ds[['A','B','C']].to_array('component').
transpose(...,'component')
```

- comps\_color\_variable (Optional[str]) The name of the xarray variable to use as the colorscale of the compositional data scatter plot. Optional, if not specified, can be customized in the GUI.
- **xmin** (*float*) Set the default q-range of the scattering data. Can be customized in the GUI
- xmax (float) Set the default q-range of the scattering data. Can be customized in the

```
GUI

    Usage

                 • ```python -
                 • DatasetWidget(ds) (widget =)
                 widget.run()
     next_button_callback(*args)
     prev_button_callback(*args)
     goto_callback(*args)
     composition_click_callback(figure, location, click)
     update_composition_plot()
     update_scattering_plot()
     update_plots()
     get_comps()
     initialize_plots(*args)
     update_colors(*args)
     apply_sel(*args)
     apply_isel(*args)
     extract_var(*args)
     combine_vars(*args)
     reset_dataset(*args)
     update_dropdowns(*args)
     update_sample_dim(*args)
     update_extract_coords(change)
     run()
class AFL.automation.shared.DatasetWidget.DatasetWidget_Model(dataset: Dataset, sample_dim: str)
```

6.2. Modules 757

**\_\_init\_\_**(dataset: Dataset, sample\_dim: str)

```
property dataset
     reset_dataset()
     split_vars()
          Heuristically try to split vars into categories
     get_non_sample_dims(var: str)
     apply_sel(kw)
     apply_isel(kw)
     combine_vars(combined_var: str, to_combine_vars: List[str])
     extract_var(extract_from_var: str, extract_from_coord: str)
     get_composition(variable)
     get_scattering(variable, index)
class AFL.automation.shared.DatasetWidget.DatasetWidget_View(initial_scatt_variables: List[str] |
                                                                       None = None,
                                                                       initial_comps_variable: str | None =
                                                                       None, initial_comps_color_variable:
                                                                       str \mid None = None)
     __init__(initial scatt variables: List[str] | None = None, initial comps variable: str | None = None,
               initial_comps_color_variable: str | None = None)
     update_colorscale(colors=None)
     update_selected(**kw)
     update_dropdowns(sample_vars=None, scatt_vars=None, comp_vars=None)
     plot_sas(x, y, name='SAS', append=False)
     plot\_comp(x, y, z=None, xname='x', yname='y', zname='z', colors=None)
     init_plots()
     init_buttons()
     init_checkboxes()
     init_dropdowns()
     init_inputs()
     run()
```

## AFL.automation.shared.DiffractionLabeler

#### **Functions**

## AFL.automation.shared.DiffractionLabeler.sqrt

AFL.automation.shared.DiffractionLabeler. $\mathbf{sqrt}(x,/)$  Return the square root of x.

### Classes

```
DiffractionLabeler(saxs_data, ...)

DiffractionLabelerModel(saxs_data, ...)

DiffractionLabelerView()

OrdinalEncoder(*[, categories, dtype, ...])

Encode categorical features as an integer array.
```

## AFL.automation.shared.DiffractionLabeler.DiffractionLabeler

Bases: object
\_\_init\_\_(saxs\_data, composition\_data, possible\_phase\_labels)

```
_init__(saxs_data, composition_data, ...)
 change_model_callback(data)
 change_norder_callback(data)
 change_qstar_callback(figure, location, click)
 draw_peaks(peaks)
 goto_callback(click)
 label(label)
 next_button_callback(click)
 prev_button_callback(click)
 run()
 ternary_click_callback(figure, location, click)
 update_plot()
__init__(saxs_data, composition_data, possible_phase_labels)
next_button_callback(click)
prev_button_callback(click)
goto_callback(click)
ternary_click_callback(figure, location, click)
update_plot()
draw_peaks(peaks)
change_qstar_callback(figure, location, click)
change_model_callback(data)
change_norder_callback(data)
label(label)
run()
```

### AFL.automation.shared.DiffractionLabeler.DiffractionLabelerModel

 ${\bf class} \ \, {\bf AFL.automation.shared.DiffractionLabeler.DiffractionLabelerModel} (saxs\_data, \\ composition\_data, \\ possible\_phase\_labels)$ 

```
Bases: object
__init__(saxs_data, composition_data, possible_phase_labels)
```

#### **Methods**

```
__init__(saxs_data, composition_data, ...)

get_peaks(model[, qstar, max_order])

init_models()

ordinal_phase_labels()

__init__(saxs_data, composition_data, possible_phase_labels)

ordinal_phase_labels()

init_models()

get_peaks(model, qstar=None, max_order=4)
```

## AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView

## **Methods**

```
__init__()

add_vertical_line(x[, y0, y1, row, col, line_kw])

remove_vertical_lines()

run(x, y, all_compositions, composition, ...)

update_plot(x, y, composition)

update_ternary_colors(colors)
```

```
__init__()

update_plot(x, y, composition)

remove_vertical_lines()

add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)

update_ternary_colors(colors)

run(x, y, all compositions, composition, models, possible phase labels)
```

#### AFL.automation.shared.DiffractionLabeler.OrdinalEncoder

Bases: OneToOneFeatureMixin, \_BaseEncoder

Encode categorical features as an integer array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are converted to ordinal integers. This results in a single column of integers (0 to n\_categories - 1) per feature.

Read more in the User Guide. For a comparison of different encoders, refer to: sphx\_glr\_auto\_examples\_preprocessing\_plot\_target\_encoder.py.

Added in version 0.20.

#### **Parameters**

- categories ('auto' or a list of array-like, default='auto') Categories (unique values) per feature:
  - 'auto': Determine categories automatically from the training data.
  - list: categories[i] holds the categories expected in the ith column. The passed categories should not mix strings and numeric values, and should be sorted in case of numeric values.

The used categories can be found in the categories\_ attribute.

- **dtype** (number type, default=np.float64) Desired dtype of output.
- handle\_unknown ({'error', 'use\_encoded\_value'}, default='error') When set to 'error' an error will be raised in case an unknown categorical feature is present during transform. When set to 'use\_encoded\_value', the encoded value of unknown categories will be set to the value given for the parameter unknown\_value. In inverse\_transform(), an unknown category will be denoted as None.

Added in version 0.24.

• unknown\_value (int or np.nan, default=None) — When the parameter handle\_unknown is set to 'use\_encoded\_value', this parameter is required and will set the

encoded value of unknown categories. It has to be distinct from the values used to encode any of the categories in *fit*. If set to np.nan, the *dtype* parameter must be a float dtype.

Added in version 0.24.

• encoded\_missing\_value (int or np.nan, default=np.nan) — Encoded value of missing categories. If set to np.nan, then the dtype parameter must be a float dtype.

Added in version 1.1.

- min\_frequency (int or float, default=None) Specifies the minimum frequency below which a category will be considered infrequent.
  - If int, categories with a smaller cardinality will be considered infrequent.
  - If float, categories with a smaller cardinality than min\_frequency \* n\_samples will be considered infrequent.

Added in version 1.3: Read more in the User Guide.

• max\_categories (int, default=None) – Specifies an upper limit to the number of output categories for each input feature when considering infrequent categories. If there are infrequent categories, max\_categories includes the category representing the infrequent categories along with the frequent categories. If None, there is no limit to the number of output features.

max\_categories do **not** take into account missing or unknown categories. Setting unknown\_value or encoded\_missing\_value to an integer will increase the number of unique integer codes by one each. This can result in up to max\_categories + 2 integer codes.

Added in version 1.3: Read more in the User Guide.

#### categories\_

The categories of each feature determined during fit (in order of the features in X and corresponding with the output of transform). This does not include categories that weren't seen during fit.

#### **Type**

list of arrays

#### n\_features\_in\_

Number of features seen during fit.

Added in version 1.0.

## Type

int

### feature\_names\_in\_

Names of features seen during fit. Defined only when *X* has feature names that are all strings.

Added in version 1.0.

#### **Type**

ndarray of shape (n features in ,)

#### infrequent\_categories\_

Defined only if infrequent categories are enabled by setting  $min\_frequency$  or  $max\_categories$  to a non-default value.  $infrequent\_categories\_[i]$  are the infrequent categories for feature i. If the feature i has no infrequent categories  $infrequent\_categories\_[i]$  is None.

Added in version 1.3.

## **Type**

list of ndarray

#### See also

#### OneHotEncoder

Performs a one-hot encoding of categorical features. This encoding is suitable for low to medium cardinality categorical variables, both in supervised and unsupervised settings.

#### TargetEncoder

Encodes categorical features using supervised signal in a classification or regression pipeline. This encoding is typically suitable for high cardinality categorical variables.

#### LabelEncoder

Encodes target labels with values between 0 and n\_classes-1.

#### **Notes**

With a high proportion of *nan* values, inferring categories becomes slow with Python versions before 3.10. The handling of *nan* values was improved from Python 3.10 onwards, (c.f. bpo-43475).

## **Examples**

Given a dataset with two features, we let the encoder find the unique values per feature and transform the data to an ordinal encoding.

By default, *OrdinalEncoder* is lenient towards missing values by propagating them.

You can use the parameter encoded\_missing\_value to encode missing values.

```
>>> enc.set_params(encoded_missing_value=-1).fit_transform(X)
array([[ 1.,  0.],

(continues on next page)
```

(continued from previous page)

```
[ 0., 1.],
[ 0., -1.]])
```

Infrequent categories are enabled by setting *max\_categories* or *min\_frequency*. In the following example, "a" and "d" are considered infrequent and grouped together into a single category, "b" and "c" are their own categories, unknown values are encoded as 3 and missing values are encoded as 4.

```
>>> X_train = np.array(
        [["a"] * 5 + ["b"] * 20 + ["c"] * 10 + ["d"] * 3 + [np.nan]],
. . .
        dtype=object).T
>>> enc = OrdinalEncoder(
        handle_unknown="use_encoded_value", unknown_value=3,
        max_categories=3, encoded_missing_value=4)
>>> _ = enc.fit(X_train)
>>> X_test = np.array([["a"], ["b"], ["c"], ["d"], ["e"], [np.nan]], dtype=object)
>>> enc.transform(X_test)
array([[2.],
       [0.],
       [1.],
       [2.],
       [3.],
       [4.]])
```

\_\_init\_\_(\*, categories='auto', dtype=<class 'numpy.float64'>, handle\_unknown='error', unknown\_value=None, encoded\_missing\_value=nan, min\_frequency=None, max\_categories=None)

#### **Methods**

| init(*[, categories, dtype,])                      |   |
|--|---|
| fit(X[,y])   | Fit the OrdinalEncoder to X.                          |
| <pre>fit_transform(X[, y])</pre>                   | Fit to data, then transform it.                       |
| <pre>get_feature_names_out([input_features])</pre> | Get output feature names for transformation.          |
| <pre>get_metadata_routing()</pre>                  | Get metadata routing of this object.                  |
| <pre>get_params([deep])</pre>                      | Get parameters for this estimator.                    |
| $inverse\_transform(X)$                            | Convert the data back to the original representation. |
| <pre>set_output(*[, transform])</pre>              | Set output container.                                 |
| <pre>set_params(**params)</pre>                    | Set the parameters of this estimator.                 |
| transform(X)                                       | Transform X to ordinal codes.                         |

### **Attributes**

infrequent\_categories\_

Infrequent categories for each feature.

```
__init__(*, categories='auto', dtype=<class 'numpy.float64'>, handle_unknown='error', unknown_value=None, encoded_missing_value=nan, min_frequency=None, max_categories=None)
```

```
fit(X, y=None)
```

Fit the OrdinalEncoder to X.

#### **Parameters**

- **X** (array-like of shape (n\_samples, n\_features)) The data to determine the categories of each feature.
- **y** (*None*) Ignored. This parameter exists only for compatibility with Pipeline.

#### Returns

self - Fitted encoder.

## Return type

object

## transform(X)

Transform X to ordinal codes.

#### **Parameters**

X(array-like of shape (n\_samples, n\_features)) - The data to encode.

#### Returns

**X\_out** – Transformed input.

#### **Return type**

ndarray of shape (n\_samples, n\_features)

## $inverse\_transform(X)$

Convert the data back to the original representation.

#### **Parameters**

 $\mathbf{X}$  (array-like of shape (n\_samples, n\_encoded\_features)) — The transformed data.

#### Returns

 $X_{tr}$  – Inverse transformed array.

## Return type

ndarray of shape (n\_samples, n\_features)

## **fit\_transform**(*X*, *y=None*, \*\*fit\_params)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit\_params* and returns a transformed version of *X*.

#### **Parameters**

- X (array-like of shape (n\_samples, n\_features)) Input samples.
- y (array-like of shape (n\_samples,) or (n\_samples, n\_outputs), default=None) Target values (None for unsupervised transformations).

• \*\*fit\_params (dict) - Additional fit parameters.

#### Returns

**X\_new** – Transformed array.

## **Return type**

ndarray array of shape (n\_samples, n\_features\_new)

### get\_feature\_names\_out(input\_features=None)

Get output feature names for transformation.

#### **Parameters**

input\_features (array-like of str or None, default=None) - Input features.

- If *input\_features* is *None*, then *feature\_names\_in\_* is used as feature names in. If *feature\_names\_in\_* is not defined, then the following input feature names are generated: ["x0", "x1", ..., "x(n\_features\_in\_ 1)"].
- If *input\_features* is an array-like, then *input\_features* must match *feature\_names\_in\_* if *feature\_names\_in\_* is defined.

#### Returns

**feature\_names\_out** – Same as input features.

#### **Return type**

ndarray of str objects

## get\_metadata\_routing()

Get metadata routing of this object.

Please check User Guide on how the routing mechanism works.

## Returns

**routing** – A MetadataRequest encapsulating routing information.

## Return type

MetadataRequest

#### get\_params(deep=True)

Get parameters for this estimator.

## **Parameters**

**deep** (*bool*, *default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

#### Returns

**params** – Parameter names mapped to their values.

### **Return type**

dict

## property infrequent\_categories\_

Infrequent categories for each feature.

```
set_output(*, transform=None)
```

Set output container.

See sphx\_glr\_auto\_examples\_miscellaneous\_plot\_set\_output.py for an example on how to use the API.

#### **Parameters**

**transform**({"default", "pandas", "polars"}, default=None)—Configure output of *transform* and *fit\_transform*.

```
• "default": Default output format of a transformer
```

- "pandas": DataFrame output
- "polars": Polars output
- None: Transform configuration is unchanged

Added in version 1.4: "polars" option was added.

#### Returns

**self** – Estimator instance.

### Return type

estimator instance

```
set_params(**params)
```

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as Pipeline). The latter have parameters of the form <component>\_\_<parameter> so that it's possible to update each component of a nested object.

#### **Parameters**

\*\*params (dict) – Estimator parameters.

#### Returns

**self** – Estimator instance.

## Return type

estimator instance

```
__init__(saxs_data, composition_data, possible_phase_labels)
next_button_callback(click)
prev_button_callback(click)
goto_callback(click)
ternary_click_callback(figure, location, click)
update_plot()
draw_peaks(peaks)
change_qstar_callback(figure, location, click)
change_model_callback(data)
change_norder_callback(data)
label(label)
run()
```

```
__init__(saxs_data, composition_data, possible_phase_labels)
    ordinal_phase_labels()
    init_models()
    get_peaks(model, qstar=None, max_order=4)

class AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView
    __init__()
    update_plot(x, y, composition)
    remove_vertical_lines()
    add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
    update_ternary_colors(colors)
    run(x, y, all_compositions, composition, models, possible_phase_labels)
```

#### AFL.automation.shared.MutableQueue

#### **Classes**

MutableQueue()

Thread-safe, mutable queue

### AFL.automation.shared.MutableQueue.MutableQueue

class AFL.automation.shared.MutableQueue.MutableQueue

Bases: object

Thread-safe, mutable queue

Unlike the standard library CPython queue, this class supportes positional inserts, deletions, and reordering. The tradeoff is performance for both reads and writes to the queue.

\_\_init\_\_()

| init()                                  |  |
|---|--|
| empty()                                 |  |
| <pre>get([loc, block, timeout])</pre>   | Get next item from queue               |
| <pre>iterationid()</pre>                |  |
| <pre>move(old_index[, new_index])</pre> | Move item in queue                     |
| put(item, loc)                          | Insert an item at the top of the queue |
| qsize()                                 |  |
| remove(loc)                             | Remove an item from the queue          |

```
__init__()
qsize()
iterationid()
empty()
put(item, loc)
    Insert an item at the top of the queue
remove(loc)
    Remove an item from the queue
get(loc=0, block=True, timeout=None)
    Get next item from queue
move(old_index, new_index=None)
    Move item in queue
```

## **Exceptions**

| Empty | Exception raised by Queue.get(block=0)/get_nowait(). |
|-------|--|
| Full  | Exception raised by Queue.put(block=0)/put_nowait(). |

## AFL.automation.shared.MutableQueue.Empty

### AFL.automation.shared.MutableQueue.Full

```
exception AFL.automation.shared.MutableQueue.Full Exception raised by Queue.put(block=0)/put_nowait().
```

class AFL.automation.shared.MutableQueue.MutableQueue

```
Thread-safe, mutable queue
```

Unlike the standard library CPython queue, this class supportes positional inserts, deletions, and reordering. The tradeoff is performance for both reads and writes to the queue.

## AFL.automation.shared.PersistentConfig

#### **Classes**

| MutableMapping()                               | A MutableMapping is a generic container for associating key/value pairs. |
|--|--|
| <pre>PersistentConfig(path[, defaults,])</pre> | A dictionary-like class that serializes changes to disk                  |

### AFL.automation.shared.PersistentConfig.MutableMapping

### class AFL.automation.shared.PersistentConfig.MutableMapping

```
Bases: Mapping

A MutableMapping is a generic container for associating key/value pairs.

This class provides concrete generic implementations of all methods except for __getitem__, __setitem__, __delitem__, __iter__, and __len__.

__init__()
```

| init()                |  |
|-----------------------|--|
| clear()               |  |
| <pre>get(k[,d])</pre> |  |
| <pre>items()</pre>    |  |
| keys()                |  |
| pop(k[,d])            | If key is not found, d is returned if given, otherwise KeyError is raised.   |
| <pre>popitem()</pre>  | as a 2-tuple; but raise KeyError if D is empty.  |
| setdefault(k[,d])     |  |
| update([E, ]**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(): $D[k] = v$ |
| values()              |  |

 $clear() \rightarrow None$ . Remove all items from D.

 $get(k[,d]) \rightarrow D[k]$  if k in D, else d. d defaults to None.

 $\textbf{items()} \rightarrow a \; \text{set-like object providing a view on D's items}$ 

**keys**()  $\rightarrow$  a set-like object providing a view on D's keys

 $\mathbf{pop}(k \big[, d \, \big]) \to \mathbf{v},$  remove specified key and return the corresponding value.

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

 $\textbf{popitem()} \rightarrow (k,\,v),$  remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

 $\mathbf{setdefault}(k \big[, d \big]) \to \mathbf{D}. \mathbf{get}(\mathbf{k}, \mathbf{d}), \text{ also set } \mathbf{D}[\mathbf{k}] \text{=-d if } \mathbf{k} \text{ not in } \mathbf{D}$ 

**update**( $[E, ]^{**}F$ )  $\rightarrow$  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

**values()**  $\rightarrow$  an object providing a view on D's values

## AFL.automation.shared.PersistentConfig.PersistentConfig

 $\textbf{class} \ \texttt{AFL}. automation. shared. Persistent Config. \textbf{Persistent Config} (\textit{path}, \textit{defaults} = \textit{None}, \\$ 

overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

Bases: MutableMapping

A dictionary-like class that serializes changes to disk

This class provides dictionary-like setters and getters (e.g., [] and .update()) but, by default, all modifications are written into a file in json format with the root keys as timestamps. Modifications can be blocked by locking the config, and writing to disk can be disabled by setting the appropriate member attributes (see constructor). On instantiation, if provided with a previously saved configuration file, PersistentConfig will load the file's contents into memory and use the more recent configuration.

\_\_init\_\_(path, defaults=None, overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

#### Constructor

#### **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

| init(path[, defaults, overrides, lock,])                  |      | Constructor  |
|---|------|--|
| clear()   |      |  |
| <pre>get(k[,d])</pre>                                     |      |  |
| <pre>get_historical_values(key[, vert_to_datetime])</pre> | con- | Convenience method for gathering historical values of a parameter          |
| items()   |      | ·  |
| keys()  |      |  |
| <i>pop</i> (k[,d])  |      | If key is not found, d is returned if given, otherwise KeyError is raised. |
| <pre>popitem()</pre>                                      |      | as a 2-tuple; but raise KeyError if D is empty.                            |
| <pre>revert([nth, datetime_key])</pre>                    |      | Revert config to a historical config                                       |
| setdefault(k[,d])   |      |  |
| toJSON()  |      | Serialize the config to json   |
| <pre>update(update_dict)</pre>                            |      | Update several values in config at once                                    |
| values()  |      |  |

\_\_init\_\_(path, defaults=None, overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

#### Constructor

#### **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

```
__getitem__(key)
```

Dictionary-like getter via config["param"]

```
__setitem__(key, value)
```

Dictionary-like setter via config["param"]=value

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

```
toJSON()
```

Serialize the config to json

```
update(update_dict)
```

Update several values in config at once

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

```
revert(nth=None, datetime_key=None)
```

Revert config to a historical config

#### **Parameters**

- **nth** (int, **optional\***) Integer index of historical value to revert to. Can be negative to count from end of history. Note that -1 will correspond to the current config.
- datetime\_key (str, optional) datetime formatted string as defined by date-time\_key\_format

```
get_historical_values(key, convert_to_datetime=False)
```

Convenience method for gathering historical values of a parameter

**clear()**  $\rightarrow$  None. Remove all items from D.

 $get(k[,d]) \rightarrow D[k]$  if k in D, else d. d defaults to None.

**items()**  $\rightarrow$  a set-like object providing a view on D's items

**keys()**  $\rightarrow$  a set-like object providing a view on D's keys

 $pop(k[,d]) \rightarrow v$ , remove specified key and return the corresponding value.

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

```
popitem() \rightarrow (k, v), remove and return some (key, value) pair
```

as a 2-tuple; but raise KeyError if D is empty.

**setdefault**(k[, d])  $\rightarrow$  D.get(k,d), also set D[k]=d if k not in D

**values()**  $\rightarrow$  an object providing a view on D's values

class AFL.automation.shared.PersistentConfig.PersistentConfig(path, defaults=None,

overrides=None, lock=False, write=True, max\_history=10000, datetime\_key\_format='%y/%d/%m %H:%M:%S.%f')

A dictionary-like class that serializes changes to disk

This class provides dictionary-like setters and getters (e.g., [] and .update()) but, by default, all modifications are written into a file in json format with the root keys as timestamps. Modifications can be blocked by locking the config, and writing to disk can be disabled by setting the appropriate member attributes (see constructor). On instantiation, if provided with a previously saved configuration file, PersistentConfig will load the file's contents into memory and use the more recent configuration.

```
__init__(path, defaults=None, overrides=None, lock=False, write=True, max_history=10000, datetime_key_format='%y/%d/%m %H:%M:%S.%f')
```

Constructor

## **Parameters**

- path (str or pathlib.Path) File path to file in which this config is or will be stored. This file will be created if it does not exist
- **defaults** (*dict*) Default values to use if no saved config is available or if parameters are missing from a saved config.
- **overrides** (*dict*) Values to use that override parameters in a saved config. These parameters can be changed after the PersistentConfig object is instantiated.
- **lock** (*bool*) If True, an AttributeError will be raised if the user attempts to modify the config
- write (bool) If False, all writing to the config file will be disabled and a warning will be emitted each time the config is modified.
- **datetime\_key\_format** (*str*) String defining the root level keys of the json-serialized file. See https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

```
__getitem__(key)
```

Dictionary-like getter via config["param"]

```
__setitem__(key, value)
```

Dictionary-like setter via config["param"]=value

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

### toJSON()

Serialize the config to json

### update(update\_dict)

Update several values in config at once

Changes will be written to PersistentConfig.path if PersistentConfig.write is True (default).

## revert(nth=None, datetime\_key=None)

Revert config to a historical config

#### **Parameters**

- **nth** (int, **optional\***) Integer index of historical value to revert to. Can be negative to count from end of history. Note that -1 will correspond to the current config.
- datetime\_key (str, optional) datetime formatted string as defined by datetime\_key\_format

```
get_historical_values(key, convert_to_datetime=False)
```

Convenience method for gathering historical values of a parameter

## AFL.automation.shared.ServerDiscovery

### **Classes**

| AsyncServiceBrowser(zeroconf, type_[,])                 | Used to browse for a service for specific type(s).            |
|---|---|
| AsyncServiceInfo  | An async version of ServiceInfo.                              |
| AsyncZeroconf([interfaces, unicast,])                   | Implementation of Zeroconf Multicast DNS Service<br>Discovery |
| AsyncZeroconfServiceTypes()                             | An async version of ZeroconfServiceTypes.                     |
| <pre>IPVersion(value[, names, module, qualname,])</pre> |   |
| RunThread(coro)   |   |
| ServerDiscovery()                                       | ServerDiscovery class   |
| <pre>ServiceBrowser(zc, type_[, handlers,])</pre>       | Used to browse for a service of a specific type.              |
| ServiceInfo   | Service information.  |
| ServiceStateChange(value[, names, module,])             |   |
| Zeroconf([interfaces, unicast, ip_version,])            | Implementation of Zeroconf Multicast DNS Service<br>Discovery |

## AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser

class AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser(zeroconf: Zeroconf, type\_: str |

list, handlers: ServiceListener |
list[Callable[[...], None]] | None
= None, listener: ServiceListener
| None = None, addr: str | None
= None, port: int = 5353, delay:
int = 10000, question\_type:
DNSQuestionType | None =
None)

Bases: \_ServiceBrowserBase

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- question\_type: The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

\_\_init\_\_(zeroconf: Zeroconf, type\_:  $str \mid list$ , handlers:  $ServiceListener \mid list[Callable[[...], None]] \mid None = None$ , listener:  $ServiceListener \mid None = None$ , addr:  $str \mid None = None$ , port: int = 5353, delay: int = 10000,  $question\_type$ :  $DNSQuestionType \mid None = None$ )  $\rightarrow$  None

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- question\_type: The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

#### **Methods**

| init(zeroconf, type_[, handlers,])                | Used to browse for a service for specific type(s).          |
|---|---|
| <pre>async_cancel()</pre>                         | Cancel the browser.   |
| <pre>async_update_records(zc, now, records)</pre> | Callback invoked by Zeroconf when new information arrives.  |
| async_update_records_complete()                   | Called when a record update has completed for all handlers. |
| <pre>update_record(zc, now, record)</pre>         | Update a single record.                                     |

#### **Attributes**

```
done
query_scheduler

types
zc
service_state_changed
```

```
__init__(zeroconf: Zeroconf, type_: str \mid list, handlers: ServiceListener \mid list[Callable[[...], None]] \mid None = None, listener: ServiceListener \mid None = None, addr: str \mid None = None, port: int = 5353, delay: int = 10000, question_type: DNSQuestionType \mid None = None) \rightarrow None
```

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- question\_type: The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

```
async async_cancel() \rightarrow None
```

Cancel the browser.

done

query\_scheduler

types

zc

async\_update\_records(zc: Zeroconf, now: float\_, records: list[RecordUpdate])

Callback invoked by Zeroconf when new information arrives.

Updates information required by browser in the Zeroconf cache.

Ensures that there is are no unnecessary duplicates in the list.

This method will be run in the event loop.

### async\_update\_records\_complete()

Called when a record update has completed for all handlers.

At this point the cache will have the new records.

This method will be run in the event loop.

This method is expected to be overridden by subclasses.

### service\_state\_changed

```
update\_record(zc: Zeroconf, now: float, record: DNSRecord) \rightarrow None
```

Update a single record.

This method is deprecated and will be removed in a future version. update\_records should be implemented instead.

## AFL.automation.shared.ServerDiscovery.AsyncServiceInfo

## class AFL.automation.shared.ServerDiscovery.AsyncServiceInfo

Bases: ServiceInfo

An async version of ServiceInfo.

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

## **Methods**

| init(*args, **kwargs)  addresses_by_version(version)  async_clear_cache()  async_request(zc, timeout[, question_type,])  async_update_records(zc, now, records)  async_update_records_complete()  async_wait(timeout[, loop])  List addresses matching IP version.  Clear the cache for this service info.  Returns true if the service could be discovered on the network, and updates this object with details discovered.  Updates service information from a DNS record.  Called when a record update has completed for a handlers.  Calling task waits for a given number of millisecond or until notified. |
|--|
| async_clear_cache()       Clear the cache for this service info.         async_request(zc, timeout[, question_type,])       Returns true if the service could be discovered on the network, and updates this object with details discovered.         async_update_records(zc, now, records)       Updates service information from a DNS record.         async_update_records_complete()       Called when a record update has completed for a handlers.         async_wait(timeout[, loop])       Calling task waits for a given number of millisecond or until notified.                                       |
| async_clear_cache()       Clear the cache for this service info.         async_request(zc, timeout[, question_type,])       Returns true if the service could be discovered on the network, and updates this object with details discovered.         async_update_records(zc, now, records)       Updates service information from a DNS record.         async_update_records_complete()       Called when a record update has completed for a handlers.         async_wait(timeout[, loop])       Calling task waits for a given number of millisecond or until notified.                                       |
| async_request(zc, timeout[, question_type,])       Returns true if the service could be discovered on the network, and updates this object with details discovered.         async_update_records(zc, now, records)       Updates service information from a DNS record.         async_update_records_complete()       Called when a record update has completed for a handlers.         async_wait(timeout[, loop])       Calling task waits for a given number of millisecond or until notified.  |
| network, and updates this object with details discovered.  async_update_records(zc, now, records)  async_update_records_complete()  async_wait(timeout[, loop])  Called when a record update has completed for a handlers.  Calling task waits for a given number of millisecond or until notified.  |
| async_update_records_complete()Called when a record update has completed for a handlers.async_wait(timeout[, loop])Calling task waits for a given number of millisecond or until notified.   |
| handlers.  Calling task waits for a given number of millisecond or until notified.   |
| or until notified.   |
|  |
| dns_addresses([override_ttl, version]) Return matching DNSAddress from ServiceInfo.  |
| dns_nsec(missing_types[, override_ttl]) Return DNSNsec from ServiceInfo.   |
| dns_pointer([override_ttl]) Return DNSPointer from ServiceInfo.  |
| dns_service([override_ttl]) Return DNSService from ServiceInfo.  |
| dns_text([override_ttl]) Return DNSText from ServiceInfo.  |
| get_address_and_nsec_records([override_ttl]) Build a set of address records and NSEC records for non-present record types.   |
| get_name() Name accessor   |
| <pre>ip_addresses_by_version(version)</pre> List ip_address objects matching IP version.   |
| <pre>load_from_cache(zc[, now])</pre> Populate the service info from the cache.  |
| parsed_addresses([version]) List addresses in their parsed string form.  |
| parsed_scoped_addresses([version]) Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with "< <interface_index> when available</interface_index>  |
| request(zc, timeout[, question_type, addr, port])  Returns true if the service could be discovered on the network, and updates this object with details discovered.  |
| set_server_if_missing() Set the server if it is missing.   |
| <pre>update_record(zc, now, record)</pre> Update a single record.  |

## **Attributes**

```
host_ttl
interface_index
key
other\_ttl
port
priority
server
server_key
text
type
weight
addresses
                                                 IPv4 addresses of this service.
decoded_properties
                                                 Return properties as strings.
                                                 The name of the service.
name
                                                 Return properties as bytes.
properties
```

```
interface_index
key
other_ttl
port
priority
server
server_key
text
type
weight
__init__(*args, **kwargs)
```

host\_ttl

#### addresses

IPv4 addresses of this service.

Only IPv4 addresses are returned for backward compatibility. Use addresses\_by\_version() or parsed\_addresses() to include IPv6 addresses as well.

### addresses\_by\_version(version: IPVersion)

List addresses matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

## async\_clear\_cache()

Clear the cache for this service info.

```
async_request(zc: Zeroconf, timeout: float, question\_type: DNSQuestionType | None = None, addr: str | None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

This method will be run in the event loop.

Passing addr and port is optional, and will default to the mDNS multicast address and port. This is useful for directing requests to a specific host that may be able to respond across subnets.

#### **Parameters**

- **zc** Zeroconf instance
- timeout time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- port port to send the request to

```
async_update_records(zc: Zeroconf, now: float_, records: list[RecordUpdate])
```

Updates service information from a DNS record.

This method will be run in the event loop.

#### async\_update\_records\_complete()

Called when a record update has completed for all handlers.

At this point the cache will have the new records.

This method will be run in the event loop.

```
async_wait(timeout: float, loop: AbstractEventLoop | None = None) <math>\rightarrow None
```

Calling task waits for a given number of milliseconds or until notified.

## decoded\_properties

Return properties as strings.

```
dns\_addresses(override\_ttl: int\_ | None = None, version: IPVersion = IPVersion.All) \rightarrow list[DNSAddress]
Return matching DNSAddress from ServiceInfo.
```

```
dns\_nsec(missing\_types: list[int], override\_ttl: int\_ | None = None) \rightarrow DNSNsec
```

Return DNSNsec from ServiceInfo.

```
dns_pointer(override\_ttl: int_ | None = None) \rightarrow DNSPointer
```

Return DNSPointer from ServiceInfo.

### $dns_service(override_ttl: int_| None = None) \rightarrow DNSService$

Return DNSService from ServiceInfo.

```
dns_text(override_ttl: int_| None = None) \rightarrow DNSText
```

Return DNSText from ServiceInfo.

### $get_address_and_nsec_records(override_ttl: int_|None = None) \rightarrow set[DNSRecord]$

Build a set of address records and NSEC records for non-present record types.

```
get_name() \rightarrow str
```

Name accessor

## ip\_addresses\_by\_version(version: IPVersion)

List ip\_address objects matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
load_from_cache(zc: Zeroconf, now: float_| None = None) \rightarrow bool
```

Populate the service info from the cache.

This method is designed to be threadsafe.

#### name

The name of the service.

```
parsed\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

List addresses in their parsed string form.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
parsed\_scoped\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

Equivalent to parsed\_addresses, with the exception that IPv6 Link-Local addresses are qualified with %<interface\_index> when available

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

#### properties

Return properties as bytes.

```
request(zc: Zeroconf, timeout: float, question_type: DNSQuestionType | None = None, addr: str \mid None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_request* cannot be completed.

#### **Parameters**

• **zc** – Zeroconf instance

- timeout time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- port port to send the request to

```
set_server_if_missing() → None
```

Set the server if it is missing.

This function is for backwards compatibility.

```
\mathbf{update\_record}(\mathit{zc} \colon \mathsf{Zeroconf}, \mathit{now} \colon \mathit{float}, \mathit{record} \colon \mathit{DNSRecord}) \to \mathsf{None}
```

Update a single record.

This method is deprecated and will be removed in a future version. update\_records should be implemented instead.

## AFL.automation.shared.ServerDiscovery.AsyncZeroconf

None)

Bases: object

Implementation of Zeroconf Multicast DNS Service Discovery

Supports registration, unregistration, queries and browsing.

The async version is currently a wrapper around Zeroconf which is now also async. It is expected that an asyncio event loop is already running before creating the AsyncZeroconf object.

```
__init__(interfaces: Sequence[str | int | tuple[tuple[str, int, int]] | InterfaceChoice =
InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool
= False, zc: Zeroconf | None = None) → None
```

Creates an instance of the Zeroconf class, establishing multicast communications, and listening.

#### **Parameters**

• **interfaces** – **InterfaceChoice** or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

| init([interfaces, unicast, ip_version,])               | Creates an instance of the Zeroconf class, establishing multicast communications, and listening.   |
|--|--|
| <pre>async_add_service_listener(type_, listener)</pre> | Adds a listener for a particular service type.   |
| async_close()  | Ends the background threads, and prevent this instance from servicing further queries.   |
| <pre>async_get_service_info(type_, name[,])</pre>      | Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds. |
| <pre>async_register_service(info[, ttl,])</pre>        | Registers service information to the network with a default TTL.   |
| <pre>async_remove_all_service_listeners()</pre>        | Removes a listener from the set that is currently listening.   |
| async_remove_service_listener(listener)                | Removes a listener from the set that is currently listening.   |
| <pre>async_unregister_all_services()</pre>             | Unregister all registered services.  |
| <pre>async_unregister_service(info)</pre>              | Unregister a service.  |
| async_update_service(info)                             | Registers service information to the network with a default TTL.   |

```
__init__(interfaces: Sequence[str | int | tuple[tuple[str, int, int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False, zc: Zeroconf | None = None) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, and listening.

#### **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

```
async async_register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating_responders: bool = False, strict: bool = True) \rightarrow Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

async async\_unregister\_all\_services()  $\rightarrow$  None

Unregister all registered services.

Unlike async\_register\_service and async\_unregister\_service, this method does not return a future and is always expected to be awaited since its only called at shutdown.

```
async async_unregister_service(info: ServiceInfo) → Awaitable
```

Unregister a service.

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

```
async async_update_service(info: ServiceInfo) → Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

```
async async_close() \rightarrow None
```

Ends the background threads, and prevent this instance from servicing further queries.

```
async async_get_service_info(type_: str, name: str, timeout: int = 3000, question_type: DNSQuestionType \mid None = None) \rightarrow AsyncServiceInfo \mid None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

#### **Parameters**

- type fully qualified service type name
- name the name of the service
- **timeout** milliseconds to wait for a response
- question\_type The type of questions to ask (DNSQuestionType.QM or DNSQuestion-Type.QU)

```
async async_add_service_listener(type_{-}: str, listener: ServiceListener) \rightarrow None
```

Adds a listener for a particular service type. This object will then have its add\_service and remove\_service methods called when services of that type become available and unavailable.

```
async async_remove_service_listener(listener: ServiceListener) \rightarrow None
```

Removes a listener from the set that is currently listening.

```
async async_remove_all_service_listeners() \rightarrow None
```

Removes a listener from the set that is currently listening.

### AFL.automation.shared.ServerDiscovery.AsyncZeroconfServiceTypes

### class AFL.automation.shared.ServerDiscovery.AsyncZeroconfServiceTypes

Bases: ZeroconfServiceTypes

An async version of ZeroconfServiceTypes.

```
\_init\_() \rightarrow None
```

Keep track of found services in a set.

| init()   | Keep track of found services in a set.                       |
|--|--|
| <pre>add_service(zc, type_, name)</pre>                | Service added.   |
| <pre>async_find([aiozc, timeout, interfaces,])</pre>   | Return all of the advertised services on any local networks. |
| <pre>find([zc, timeout, interfaces, ip_version])</pre> | Return all of the advertised services on any local networks. |
| <pre>remove_service(zc, type_, name)</pre>             | Service removed.   |
| <pre>update_service(zc, type_, name)</pre>             | Service updated.   |

```
async classmethod async_find(aiozc: AsyncZeroconf | None = None, timeout: int | float = 5, interfaces: Sequence[str | int | tuple[tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, <math>ip\_version: IPVersion | None = None) \rightarrow tuple[str, ...]
```

Return all of the advertised services on any local networks.

#### **Parameters**

- aiozc AsyncZeroconf() instance. Pass in if already have an instance running or if nondefault interfaces are needed
- timeout seconds to wait for any responses
- interfaces interfaces to listen on.
- **ip\_version** IP protocol version to use.

#### Returns

tuple of service type strings

```
__init__() \rightarrow None
```

Keep track of found services in a set.

```
add_service(zc: Zeroconf, type_{-}: str, name: str) \rightarrow None
```

Service added.

```
classmethod find(zc: Zeroconf | None = None, timeout: int | float = 5, interfaces: Sequence[str | int | tuple[tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, ip_version: IPVersion | None = None) <math>\rightarrow tuple[str, ...]
```

Return all of the advertised services on any local networks.

## **Parameters**

- **zc** Zeroconf() instance. Pass in if already have an instance running or if non-default interfaces are needed
- **timeout** seconds to wait for any responses
- **interfaces** interfaces to listen on.
- **ip\_version** IP protocol version to use.

## Returns

tuple of service type strings

```
remove_service(zc: Zeroconf, type_{-}: str, name: str) \rightarrow None Service removed.
```

```
update_service(zc: Zeroconf, type_-: str, name: str) \rightarrow None Service updated.
```

## AFL.automation.shared.ServerDiscovery.IPVersion

```
Bases: Enum
__init__(*args, **kwds)
```

#### **Attributes**

```
V40nly
V60nly
All
```

```
V4Only = 1
V6Only = 2
All = 3
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
Return the member matching name.
classmethod __iter__()
Return members in definition order.
```

classmethod \_\_len\_\_()
 Return the number of members (no aliases)

## AFL.automation.shared.ServerDiscovery.RunThread

# \_\_init\_\_(coro)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

| init(coro)                 | This constructor should always be called with keyword arguments. |
|----------------------------|--|
| <pre>getName()</pre>       | Return a string used for identification purposes only.           |
| isDaemon()                 | Return whether this thread is a daemon.                          |
| is_alive()                 | Return whether the thread is alive.                              |
| <pre>join([timeout])</pre> | Wait until the thread terminates.                                |
| run()                      | Method representing the thread's activity.                       |
| setDaemon(daemonic)        | Set whether this thread is a daemon.                             |
| setName(name)              | Set the name string for this thread.                             |
| start()                    | Start the thread's activity.                                     |

#### **Attributes**

| daemon    | A boolean value indicating whether this thread is a daemon thread.            |
|-----------|---|
| ident     | Thread identifier of this thread or None if it has not been started.          |
| name      | A string used for identification purposes only.                               |
| native_id | Native integral thread ID of this thread, or None if it has not been started. |

## \_\_init\_\_(coro)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

## is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

## setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

## AFL.automation.shared.ServerDiscovery.ServerDiscovery

## class AFL.automation.shared.ServerDiscovery.ServerDiscovery

Bases: object

ServerDiscovery class

This class is used to discover AFL servers on the network using zeroconf requests that match a particular specification.

\_\_init\_\_()

#### **Methods**

| init()  |   |
|---|---|
| <pre>aio_find_server_by_name(service_name)</pre>          |   |
| discover_server_by_name(service_name)                     | Does a zeroconf request to find a named AFL-automation server on the network.               |
| <pre>find_server_by_name(service_name)</pre>              | Disambiguator for either matching or discovering a specific server by name                  |
| <pre>find_server_by_partial_name(service_name)</pre>      | Looks through the registry of discovered services for a partial name match.                 |
| <pre>find_server_by_property_match(property_name)</pre>   | Looks through the registry of discovered services for a partial match in a property string. |
| <pre>get_service_info(zeroconf, service_type, name)</pre> |   |
| <pre>manage_service_info_to_list(zeroconf,)</pre>         |   |
| <pre>match_server_by_name(service_name)</pre>             | Looks through the registry of discovered services for an exact name match.                  |
| <pre>on_service_state_change(zeroconf,)</pre>             |   |
| sa_aio_discover_server_by_name(service_name               | Does a zeroconf request to find a named AFL-automation server on the network.               |
| <pre>sa_discover_server_by_name(service_name)</pre>       | Does a zeroconf request to find a named AFL-automation server on the network.               |

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

```
async classmethod sa_aio_discover_server_by_name(service_name)
```

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

# classmethod sa\_discover\_server\_by\_name(service\_name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

#### find\_server\_by\_name(service\_name)

Disambiguator for either matching or discovering a specific server by name

#### match\_server\_by\_name(service\_name)

Looks through the registry of discovered services for an exact name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

#### find\_server\_by\_partial\_name(service\_name)

Looks through the registry of discovered services for a partial name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

## find\_server\_by\_property\_match(property\_name, property\_value)

Looks through the registry of discovered services for a partial match in a property string. Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## AFL.automation.shared.ServerDiscovery.ServiceBrowser

**class** AFL.automation.shared.ServerDiscovery.**ServiceBrowser**(zc: Zeroconf, type\_: str | list, handlers:

ServiceListener | list[Callable[..., None]] | None = None, listener: ServiceListener | None = None, addr: str | None = None, port: int = 5353, delay: int = 10000, question\_type: DNSQuestionType | None = None)

Bases: \_ServiceBrowserBase, Thread

Used to browse for a service of a specific type.

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

\_\_init\_\_(zc: Zeroconf, type\_: str | list, handlers: ServiceListener | list[Callable[..., None]] | None = None, listener: ServiceListener | None = None, addr: str | None = None, port: int = 5353, delay: int = 10000, question\_type: DNSQuestionType | None = None) → None

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- $\bullet \ \ \textit{question\_type}. \ \ \text{The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)}$

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

## **Methods**

| init(zc, type_[, handlers, listener,])            | Used to browse for a service for specific type(s).     |
|---|--|
| <pre>async_update_records(zc, now, records)</pre> | Callback invoked by Zeroconf when new information      |
|   | arrives.   |
| <pre>async_update_records_complete()</pre>        | Called when a record update has completed for all      |
|   | handlers.  |
| cancel()  | Cancel the browser.                                    |
| <pre>getName()</pre>                              | Return a string used for identification purposes only. |
| isDaemon()  | Return whether this thread is a daemon.                |
| is_alive()  | Return whether the thread is alive.                    |
| <pre>join([timeout])</pre>                        | Wait until the thread terminates.                      |
| run()   | Run the browser thread.                                |
| setDaemon(daemonic)                               | Set whether this thread is a daemon.                   |
| setName(name)                                     | Set the name string for this thread.                   |
| start()   | Start the thread's activity.                           |
| <pre>update_record(zc, now, record)</pre>         | Update a single record.                                |

## **Attributes**

| done                  |   |
|-----------------------|---|
| query_scheduler       |   |
| types                 |   |
| ZC                    |   |
| daemon                | A boolean value indicating whether this thread is a daemon thread.            |
| ident                 | Thread identifier of this thread or None if it has not been started.          |
| name                  | A string used for identification purposes only.                               |
| native_id             | Native integral thread ID of this thread, or None if it has not been started. |
| service_state_changed |   |

## done

query\_scheduler

types

zc

794

```
__init__(zc: Zeroconf, type_: str | list, handlers: ServiceListener | list[Callable[..., None]] | None = None, listener: ServiceListener | None = None, addr: str | None = None, port: int = 5353, delay: int = 10000, question_type: DNSQuestionType | None = None) \rightarrow None
```

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- question\_type: The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

## async\_update\_records(zc: Zeroconf, now: float\_, records: list[RecordUpdate])

Callback invoked by Zeroconf when new information arrives.

Updates information required by browser in the Zeroconf cache.

Ensures that there is are no unnecessary duplicates in the list.

This method will be run in the event loop.

### $async\_update\_records\_complete() \rightarrow None$

Called when a record update has completed for all handlers.

At this point the cache will have the new records.

This method will be run in the event loop.

### $cancel() \rightarrow None$

Cancel the browser.

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

## property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

## property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### $run() \rightarrow None$

Run the browser thread.

#### service\_state\_changed

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

#### $update\_record(zc: Zeroconf, now: float, record: DNSRecord) \rightarrow None$

Update a single record.

This method is deprecated and will be removed in a future version. update\_records should be implemented instead.

## AFL.automation.shared.ServerDiscovery.ServiceInfo

#### class AFL.automation.shared.ServerDiscovery.ServiceInfo

Bases: RecordUpdateListener

Service information.

Constructor parameters are as follows:

- type\_: fully qualified service type name
- name: fully qualified service name
- port: port that the service runs on
- weight: weight of the service
- priority: priority of the service
- *properties*: dictionary of properties (or a bytes object holding the contents of the *text* field). converted to str and then encoded to bytes using UTF-8. Keys with *None* values are converted to value-less attributes.
- *server*: fully qualified name for service host (defaults to name)
- host\_ttl: ttl used for A/SRV records
- other\_ttl: ttl used for PTR/TXT records
- addresses and parsed\_addresses: List of IP addresses (either as bytes, network byte order, or in parsed form as text; at most one of those parameters can be provided)
- interface\_index: scope\_id or zone\_id for IPv6 link-local addresses i.e. an identifier of the interface where the peer is connected to

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

# Methods

| init(*args, **kwargs)  |   |
|--|---|
| addresses_by_version(version)                                | List addresses matching IP version.   |
| async_clear_cache()  | Clear the cache for this service info.  |
| async_request(zc, timeout[, question_type,])                 | Returns true if the service could be discovered on the                        |
| async_requese(ze, ameout[, question_type,])                  | network, and updates this object with details discovered.                     |
| <pre>async_update_records(zc, now, records)</pre>            | Updates service information from a DNS record.                                |
| <pre>async_update_records_complete()</pre>                   | Called when a record update has completed for all handlers.                   |
| <pre>async_wait(timeout[, loop])</pre>                       | Calling task waits for a given number of milliseconds or until notified.      |
| <pre>dns_addresses([override_ttl, version])</pre>            | Return matching DNSAddress from ServiceInfo.                                  |
| <pre>dns_nsec(missing_types[, override_ttl])</pre>           | Return DNSNsec from ServiceInfo.  |
| <pre>dns_pointer([override_ttl])</pre>                       | Return DNSPointer from ServiceInfo.   |
| <pre>dns_service([override_ttl])</pre>                       | Return DNSService from ServiceInfo.   |
| <pre>dns_text([override_ttl])</pre>                          | Return DNSText from ServiceInfo.  |
| <pre>get_address_and_nsec_records([override_ttl])</pre>      | Build a set of address records and NSEC records for non-present record types. |
| <pre>get_name()</pre>  | Name accessor   |
| <pre>ip_addresses_by_version(version)</pre>                  | List ip_address objects matching IP version.                                  |
| <pre>load_from_cache(zc[, now])</pre>                        | Populate the service info from the cache.                                     |
| <pre>parsed_addresses([version])</pre>                       | List addresses in their parsed string form.                                   |
| <pre>parsed_scoped_addresses([version])</pre>                | Equivalent to parsed_addresses, with the exception                            |
|  | that IPv6 Link-Local addresses are qualified with                             |
|  | % <interface_index> when available</interface_index>                          |
| <pre>request(zc, timeout[, question_type, addr, port])</pre> | Returns true if the service could be discovered on the                        |
|  | network, and updates this object with details discov-                         |
|  | ered.   |
| <pre>set_server_if_missing()</pre>                           | Set the server if it is missing.  |
| <pre>update_record(zc, now, record)</pre>                    | Update a single record.   |

## **Attributes**

```
host_ttl
interface_index
key
other\_ttl
port
priority
server
server_key
text
type
weight
addresses
                                                 IPv4 addresses of this service.
decoded_properties
                                                 Return properties as strings.
                                                 The name of the service.
name
                                                 Return properties as bytes.
properties
```

```
host_ttl
interface_index
key
other_ttl
port
priority
server
server_key
text
type
weight
__init__(*args, **kwargs)
```

#### addresses

IPv4 addresses of this service.

Only IPv4 addresses are returned for backward compatibility. Use addresses\_by\_version() or parsed\_addresses() to include IPv6 addresses as well.

## addresses\_by\_version(version: IPVersion)

List addresses matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

#### async\_clear\_cache()

Clear the cache for this service info.

```
async_request(zc: Zeroconf, timeout: float, question\_type: DNSQuestionType | None = None, addr: str | None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

This method will be run in the event loop.

Passing addr and port is optional, and will default to the mDNS multicast address and port. This is useful for directing requests to a specific host that may be able to respond across subnets.

#### **Parameters**

- **zc** Zeroconf instance
- timeout time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- port port to send the request to

```
async_update_records(zc: Zeroconf, now: float_, records: list[RecordUpdate])
```

Updates service information from a DNS record.

This method will be run in the event loop.

#### async\_update\_records\_complete()

Called when a record update has completed for all handlers.

At this point the cache will have the new records.

This method will be run in the event loop.

```
async_wait(timeout: float, loop: AbstractEventLoop | None = None) <math>\rightarrow None
```

Calling task waits for a given number of milliseconds or until notified.

# decoded\_properties

Return properties as strings.

```
dns\_addresses(override\_ttl: int\_ | None = None, version: IPVersion = IPVersion.All) \rightarrow list[DNSAddress]
Return matching DNSAddress from ServiceInfo.
```

```
dns\_nsec(missing\_types: list[int], override\_ttl: int\_ | None = None) \rightarrow DNSNsec
```

Return DNSNsec from ServiceInfo.

```
dns_pointer(override\_ttl: int_ | None = None) \rightarrow DNSPointer
```

Return DNSPointer from ServiceInfo.

#### $dns_service(override_ttl: int_| None = None) \rightarrow DNSService$

Return DNSService from ServiceInfo.

```
dns_text(override_ttl: int_| None = None) \rightarrow DNSText
```

Return DNSText from ServiceInfo.

#### $get_address_and_nsec_records(override_ttl: int_|None = None) \rightarrow set[DNSRecord]$

Build a set of address records and NSEC records for non-present record types.

```
get_name() \rightarrow str
```

Name accessor

## ip\_addresses\_by\_version(version: IPVersion)

List ip\_address objects matching IP version.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
load_from_cache(zc: Zeroconf, now: float_ | None = None) \rightarrow bool
```

Populate the service info from the cache.

This method is designed to be threadsafe.

#### name

The name of the service.

```
parsed\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

List addresses in their parsed string form.

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

```
parsed\_scoped\_addresses(version: IPVersion = IPVersion.All) \rightarrow list[str]
```

Equivalent to parsed\_addresses, with the exception that IPv6 Link-Local addresses are qualified with %<interface\_index> when available

Addresses are guaranteed to be returned in LIFO (last in, first out) order with IPv4 addresses first and IPv6 addresses second.

This means the first address will always be the most recently added address of the given IP version.

#### properties

Return properties as bytes.

```
request(zc: Zeroconf, timeout: float, question_type: DNSQuestionType | None = None, addr: str \mid None = None, port: int = 5353) \rightarrow bool
```

Returns true if the service could be discovered on the network, and updates this object with details discovered.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_request* cannot be completed.

#### **Parameters**

• **zc** – Zeroconf instance

- **timeout** time in milliseconds to wait for a response
- question\_type question type to ask
- addr address to send the request to
- **port** port to send the request to

```
set_server_if_missing() → None
```

Set the server if it is missing.

This function is for backwards compatibility.

```
\textbf{update\_record}(\textit{zc: Zeroconf}, \textit{now: float, record: DNSRecord}) \rightarrow \texttt{None}
```

Update a single record.

This method is deprecated and will be removed in a future version. update\_records should be implemented instead.

# AFL.automation.shared.ServerDiscovery.ServiceStateChange

```
 \textbf{class} \  \, \textbf{AFL}. \textbf{automation}. \textbf{shared}. \textbf{ServerDiscovery}. \textbf{ServiceStateChange}(\textit{value}, \textit{names} = None, *, \\ \textit{module} = None, \textit{qualname} = None, \\ \textit{type} = None, \textit{start} = 1, \\ \textit{boundary} = None)
```

```
Bases: Enum
__init__(*args, **kwds)
```

# **Attributes**

```
Added

Removed

Updated
```

```
Added = 1
```

Removed = 2

Updated = 3

```
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching name.

```
classmethod __iter__()
```

Return members in definition order.

#### classmethod \_\_len\_\_()

Return the number of members (no aliases)

## AFL.automation.shared.ServerDiscovery.Zeroconf

class AFL.automation.shared.ServerDiscovery.Zeroconf(interfaces: Sequence[str | int | tuple[tuple[str,

int, int], int]] | InterfaceChoice =
InterfaceChoice.All, unicast: bool = False,
ip\_version: IPVersion | None = None,
apple\_p2p: bool = False)

Bases: QuietLogger

Implementation of Zeroconf Multicast DNS Service Discovery

Supports registration, unregistration, queries and browsing.

```
__init__(interfaces: Sequence[str | int | tuple[tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

#### **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

# **Methods**

| init([interfaces, unicast, ip_version,])                | Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.                                 |
|---|--|
| <pre>add_listener(listener, question)</pre>             | Adds a listener for a given question.  |
| <pre>add_service_listener(type_, listener)</pre>        | Adds a listener for a particular service type.   |
| <pre>async_add_listener(listener, question)</pre>       | Adds a listener for a given question.  |
| <pre>async_check_service(info, allow_name_change)</pre> | Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique.   |
| <pre>async_get_service_info(type_, name[,])</pre>       | Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds. |
| <pre>async_notify_all()</pre>                           | Schedule an async_notify_all.  |
| async_register_service(info[, ttl,])                    | Registers service information to the network with a default TTL.   |

continues on next page

Table 60 – continued from previous page

| async_remove_listener(listener) async_send(out[, addr, port, v6_flow_scope,]) async_unregister_all_services() async_unregister_service(info)  async_undate_service(info)  async_wait(timeout)  async_wait_for_start([timeout])  close()  close()  close()  generate_service_upery(info) generate_service_upery(info) generate_unregister_all_services()  generate_unregister_all_services()  generate_unregister_all_services()  generate_unregister_all_services()  generate_unregister_all_services()  generate_unregister_service_duery(info) generate_unregister_service(info, til,))  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  Notifies all waiting threads and notify listeners. Registers service information to the network with a default TTL.  Removes a listener from the set that is currently listening.  Removes a listener from the set that is currently listening.  Removes a listener from the set that is currently listening.  Send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_services() unregister all registered services.  Unregister information to the network with a default TTL.  | Table 60 - Continue  | a nom previous page                                 |
|--|--|---|
| async_unregister_all_service(info) async_update_service(info)  async_wait(timeout)  async_wait(timeout)  async_wait_for_start([timeout])  calling task waits for a given number of milliseconds or until notified.  Wait for start up for actions that require a running Zeroconf instance.  Ends the background threads, and prevent this instance from servicing further queries.  generate_service_query(info) generate_service_query(info) generate_unregister_all_services()  get_service_info(type_, name[, timeout,])  log_exception_debug(*logger_data)  log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()  register_service_listeners()  remove_all_service_listeners()  remove_listener(listener)  remove_service_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  send(out[, addr, port, v6_flow_scope, transport])  send(out[, service information to the network with a network with a network service information to the network with a network service.  Unregister_aservice(info)  unregister_all_services()  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  Registers service information to the network with a | async_remove_listener(listener)                              | Removes a listener.                                 |
| async_update_service(info) async_wait(timeout)  async_wait(timeout)  async_wait_for_start([timeout])  async_wait_for_start([timeout])  async_wait_for_start([timeout])  async_wait_for_start([timeout])  async_wait_for_start([timeout])  async_wait_for_start([timeout])  async_wait_for_start([timeout])  wait for start up for actions that require a running Zeroconf instance.  Ends the background threads, and prevent this instance from servicing further queries.  Generate a broadcast to announce a service.  Generate a possible to announce a service.  Generate a DNSOutgoing goodbye for all services and remove them from the registry.  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  log_exception_debug(*logger_data)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()  register_service(info[, ttl,])  Registers service information to the network with a default TTL.  remove_all_service_listeners()  Removes a listener from the set that is currently listening.  remove_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  send(out[, addr, port, v6_flow_scope, transport])  send(out[, addr, port, v6_flow_scope, transport])  send(out], service(info)  unregister_all_services()  unregister_service(info)  |  |   |
| async_wait(timeout)  async_wait(timeout)  async_wait(timeout)  async_wait_(timeout)  async_wait_(timeout)  async_wait_(timeout)  close()  close()  generate_service_broadcast(info, ttl[,])  generate_service_query(info)  generate_unregister_all_services()  get_service_info(type_, name[, timeout,])  log_exception_debug(*logger_data)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()  register_service_listener()  remove_all_service_listener()  send(out[, addr, port, v6_flow_scope, transport])  saync_wait (timeout)  Calling task waits for a given number of milliseconds or until notified.  Wait for start up for actions that require a running Zeroconf instance.  Generate a broadcast to announce a service.  Generate a prosecust o announce a service.  Generate a DNSOutgoing goodbye for all services and remove them from the registry.  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  Notifies all waiting threads and notify listeners.  Registers service information to the network with a default TTL.  remove_all_service_listeners()  Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_all_services()  unregister_all_services()  unregister_all_services()  unregister_aservice information to the network with a Registers service information to the network with a Registers serv     |  |   |
| default TTL.  Calling task waits for a given number of milliseconds or until notified.  Wait for start up for actions that require a running Zerocof instance.  Ends the background threads, and prevent this instance from servicing further queries.  Generate a broadcast to announce a service.  Generate a DNSOutgoing goodbye for all services and remove them from the registry.  Get_service_info(type_, name[, timeout,])  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  Registers service information to the network with a default TTL.  Remove_all_service_listeners()  remove_all_service_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service information to the network with a network with a network with a network with a network and service.  Unregister service.  Unregister all_services.  unregister_service(info)  unregister_service(info)  Registers service information to the network with a net | <pre>async_unregister_service(info)</pre>                    | Unregister a service.                               |
| or until notified.  Wait for start up for actions that require a running Zeroconf instance.  Ends the background threads, and prevent this instance from servicing further queries.  Generate _service_query(info)  generate_service_query(info)  generate_unregister_all_services()  get_service_info(type_, name[, timeout,])  log_exception_debug(*logger_data)  log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_exception_interior in the set that is currently listening.  remove_all_service_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_service(info)  Registers service information to the network with a default TTL.  Removes a listener.  Removes a listener.  Removes a listener.  Removes a listener.  Removes a listener from the set that is currently listening.  Sends an outgoing packet threadsafe.  Start Zeroconf.  Unregister all registered services.  Unregister a service.  unregister_service(info)  Registers service information to the network with a   | async_update_service(info)                                   |   |
| close()  Ends the background threads, and prevent this instance from servicing further queries.  generate_service_broadcast(info, ttl[,])  generate_service_query(info)  generate_unregister_all_services()  get_service_info(type_, name[, timeout,])  get_service_info(type_, name[, timeout,])  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  log_exception_debug(*logger_data)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()  register_service(info[, ttl,])  Registers service information to the network with a default TTL.  remove_all_service_listeners()  Removes a listener from the set that is currently listening.  remove_service_listener(listener)  Removes a listener.  Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_all_services()  unregister_service(info)  Unregister all registered services.  unregister_service(info)  Unregister a service.  Registers service information to the network with a Registers service information to the network with a Registers service information to the network with a Removes a listener.  Removes a listener.  Removes a listener from the set that is currently listening.  Sends an outgoing packet threadsafe.  Start Zeroconf.  Unregister all registered services.  Unregister all registered services.  Unregister as service.  Unregister as service.   | async_wait(timeout)  |   |
| stance from servicing further queries.  generate_service_duery(info)   | <pre>async_wait_for_start([timeout])</pre>                   |   |
| generate_service_query(info) generate_unregister_all_services()  get_service_info(type_, name[, timeout,])  get_service_info(type_, name[, timeout,])  log_exception_debug(*logger_data)  log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()  register_service(info[, ttl,])  remove_all_service_listeners()  remove_listener(listener)  remove_service_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister_service(info)  unregister a service.  Registers service information to the network with a default TTL.  Removes a listener from the set that is currently listening.  Sends an outgoing packet threadsafe.  Start Zeroconf.  unregister_service(info)  unregister all registered services.  unregister_service(info)  unregister a service.  Registers service information to the network with a   | close()  |   |
| generate_unregister_all_services       Generate a DNSOutgoing goodbye for all services and remove them from the registry.         get_service_info(type_, name[, timeout,])       Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.         log_exception_debug(*logger_data)       log_exception_once(exc, *args)         log_exception_warning(*logger_data)       Notifies all waiting threads and notify listeners.         register_service(info[, ttl,])       Registers service information to the network with a default TTL.         remove_all_service_listeners()       Removes a listener from the set that is currently listening.         remove_listener(listener)       Removes a listener.         remove_service_listener(listener)       Removes a listener from the set that is currently listening.         send(out[, addr, port, v6_flow_scope, transport])       Sends an outgoing packet threadsafe.         start()       Start Zeroconf.         unregister_all_services()       Unregister all registered services.         unregister_service(info)       Unregister as service.         update_service(info)       Registers service information to the network with a  | <pre>generate_service_broadcast(info, ttl[,])</pre>          | Generate a broadcast to announce a service.         |
| and remove them from the registry.  Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  log_exception_debug(*logger_data)  log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all()     register_service(info[, ttl,])     Registers service information to the network with a default TTL.  remove_all_service_listeners()     Removes a listener from the set that is currently listening.  remove_service_listener(listener)     Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport])     Sends an outgoing packet threadsafe.  start()     unregister_all_services()     unregister_all_services()     unregister_service(info)     unregister_service(info)     unregister as revice.  Registers service information to the network with a   | <pre>generate_service_query(info)</pre>                      | Generate a query to lookup a service.               |
| name and type, or None if no service matches by the timeout, which defaults to 3 seconds.  log_exception_debug(*logger_data)  log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all() Notifies all waiting threads and notify listeners.  register_service(info[, ttl,]) Registers service information to the network with a default TTL.  remove_all_service_listeners() Removes a listener from the set that is currently listening.  remove_service_listener(listener) Removes a listener.  remove_service_listener(listener) Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport]) Sends an outgoing packet threadsafe.  start() Start Zeroconf.  unregister_all_services() Unregister all registered services.  unregister_service(info) Unregister a service.  update_service(info) Registers service information to the network with a   | <pre>generate_unregister_all_services()</pre>                |   |
| log_exception_once(exc, *args)  log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all() Notifies all waiting threads and notify listeners. register_service(info[, ttl,]) Registers service information to the network with a default TTL. remove_all_service_listeners() Removes a listener from the set that is currently listening. remove_service_listener(listener) Removes a listener. remove_service_listener(listener) Removes a listener from the set that is currently listening. send(out[, addr, port, v6_flow_scope, transport]) Sends an outgoing packet threadsafe. start() Start Zeroconf. unregister_all_services() Unregister all registered services. unregister_service(info) Unregister a service. update_service(info) Registers service information to the network with a  | <pre>get_service_info(type_, name[, timeout,])</pre>         | name and type, or None if no service matches by the |
| log_exception_warning(*logger_data)  log_warning_once(*args)  notify_all() Notifies all waiting threads and notify listeners.  register_service(info[, ttl,]) Registers service information to the network with a default TTL.  remove_all_service_listeners() Removes a listener from the set that is currently listening.  remove_service_listener(listener) Removes a listener.  remove_service_listener(listener) Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport]) Sends an outgoing packet threadsafe.  start() Start Zeroconf.  unregister_all_services() Unregister all registered services.  unregister_service(info) Unregister a service.  update_service(info) Registers service information to the network with a   | log_exception_debug(*logger_data)                            |   |
| log_warning_once(*args)         notify_all()       Notifies all waiting threads and notify listeners.         register_service(info[, ttl,])       Registers service information to the network with a default TTL.         remove_all_service_listeners()       Removes a listener from the set that is currently listening.         remove_listener(listener)       Removes a listener.         remove_service_listener(listener)       Removes a listener from the set that is currently listening.         send(out[, addr, port, v6_flow_scope, transport])       Sends an outgoing packet threadsafe.         start()       Start Zeroconf.         unregister_all_services()       Unregister all registered services.         unregister_service(info)       Unregister a service.         update_service(info)       Registers service information to the network with a  | <pre>log_exception_once(exc, *args)</pre>                    |   |
| notify_all()Notifies all waiting threads and notify listeners.register_service(info[, ttl,])Registers service information to the network with a default TTL.remove_all_service_listeners()Removes a listener from the set that is currently listening.remove_listener(listener)Removes a listener.remove_service_listener(listener)Removes a listener from the set that is currently listening.send(out[, addr, port, v6_flow_scope, transport])Sends an outgoing packet threadsafe.start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a   | <pre>log_exception_warning(*logger_data)</pre>               |   |
| register_service(info[, ttl,])Registers service information to the network with a default TTL.remove_all_service_listeners()Removes a listener from the set that is currently listening.remove_listener(listener)Removes a listener.remove_service_listener(listener)Removes a listener from the set that is currently listening.send(out[, addr, port, v6_flow_scope, transport])Sends an outgoing packet threadsafe.start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a   | log_warning_once(*args)                                      |   |
| default TTL.  remove_all_service_listeners()  remove_listener(listener)  remove_service_listener(listener)  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_all_services()  unregister_service(info)  default TTL.  Removes a listener from the set that is currently listening.  Sendo an outgoing packet threadsafe.  Start Zeroconf.  Unregister all registered services.  Unregister a service.  update_service(info)  Registers service information to the network with a  | notify_all()   | Notifies all waiting threads and notify listeners.  |
| tening.  remove_listener(listener)  Removes a listener.  remove_service_listener(listener)  Removes a listener from the set that is currently listening.  send(out[, addr, port, v6_flow_scope, transport])  Sends an outgoing packet threadsafe.  start()  start Zeroconf.  unregister_all_services()  unregister_all_service(info)  Unregister a service.  update_service(info)  Registers service information to the network with a   | register_service(info[, ttl,])                               |   |
| remove_service_listener(listener)Removes a listener from the set that is currently listening.send(out[, addr, port, v6_flow_scope, transport])Sends an outgoing packet threadsafe.start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a   | remove_all_service_listeners()                               | ·   |
| tening.  send(out[, addr, port, v6_flow_scope, transport])  start()  unregister_all_services()  unregister_service(info)  update_service(info)  tening.  Sends an outgoing packet threadsafe.  Start Zeroconf.  Unregister all registered services.  Unregister a service.  Registers service information to the network with a  |  | Removes a listener.                                 |
| send(out[, addr, port, v6_flow_scope, transport])Sends an outgoing packet threadsafe.start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a  | remove_service_listener(listener)                            | ·   |
| start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a   | <pre>send(out[, addr, port, v6_flow_scope, transport])</pre> | Sends an outgoing packet threadsafe.                |
| unregister_service(info)Unregister a service.update_service(info)Registers service information to the network with a   |  |   |
| update_service(info)  Registers service information to the network with a  | unregister_all_services()                                    | Unregister all registered services.                 |
|  | unregister_service(info)                                     |   |
|  | <pre>update_service(info)</pre>                              |   |

#### **Attributes**

listeners

started Check if the instance has started.

```
__init__(interfaces: Sequence[str | int | tuple[tuple[str, int, int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

#### **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for InterfaceChoice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

#### property started: bool

Check if the instance has started.

```
start() \rightarrow None
```

Start Zeroconf.

```
async async_wait_for_start(timeout: float = 9) \rightarrow None
```

Wait for start up for actions that require a running Zeroconf instance.

Throws NotRunningException if the instance is not running or could not be started.

```
property listeners: set[RecordUpdateListener]
```

```
async async_wait(timeout: float) \rightarrow None
```

Calling task waits for a given number of milliseconds or until notified.

```
notify_all() \rightarrow None
```

Notifies all waiting threads and notify listeners.

```
async\_notify\_all() \rightarrow None
```

Schedule an async\_notify\_all.

```
\texttt{get\_service\_info}(type\_: str, name: str, timeout: int = 3000, question\_type: DNSQuestionType | None = None) <math>\rightarrow ServiceInfo | None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

#### **Parameters**

• **type** – fully qualified service type name

- name the name of the service
- timeout milliseconds to wait for a response
- **question\_type** The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

```
add\_service\_listener(type\_: str, listener: ServiceListener) \rightarrow None
```

Adds a listener for a particular service type. This object will then have its add\_service and remove\_service methods called when services of that type become available and unavailable.

```
remove\_service\_listener(listener: ServiceListener) \rightarrow None
```

Removes a listener from the set that is currently listening.

```
remove\_all\_service\_listeners() \rightarrow None
```

Removes a listener from the set that is currently listening.

```
register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating responders: bool = False, strict: bool = True) \rightarrow None
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *register\_service* cannot be completed.

```
async async_register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating\_responders: bool = False, strict: bool = True) \rightarrow Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

```
update\_service(info: ServiceInfo) \rightarrow None
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_update\_service* cannot be completed.

```
async async_update_service(info: ServiceInfo) → Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

```
async async_get_service_info(type_: str, name: str, timeout: int = 3000, question_type: DNSQuestionType | None = None) <math>\rightarrow AsyncServiceInfo | None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

## **Parameters**

- type fully qualified service type name
- name the name of the service
- timeout milliseconds to wait for a response
- question\_type The type of questions to ask (DNSQuestionType.QM or DNSQuestion-Type.QU)

 $\begin{tabular}{ll} \begin{tabular}{ll} \beg$ 

Generate a broadcast to announce a service.

 $generate\_service\_query(info: ServiceInfo) \rightarrow DNSOutgoing$ 

Generate a query to lookup a service.

**unregister\_service**(*info:* ServiceInfo) → None

Unregister a service.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_unregister\_service* cannot be completed.

async async\_unregister\_service(info: ServiceInfo)  $\rightarrow$  Awaitable

Unregister a service.

 $generate\_unregister\_all\_services() \rightarrow DNSOutgoing | None$ 

Generate a DNSOutgoing goodbye for all services and remove them from the registry.

async async\_unregister\_all\_services()  $\rightarrow$  None

Unregister all registered services.

Unlike async\_register\_service and async\_unregister\_service, this method does not return a future and is always expected to be awaited since its only called at shutdown.

 $unregister\_all\_services() \rightarrow None$ 

Unregister all registered services.

While it is not expected during normal operation, this function may raise EventLoopBlocked if the underlying call to *async\_unregister\_all\_services* cannot be completed.

**async\_check\_service**(info: ServiceInfo, allow\_name\_change: bool, cooperating\_responders: bool = False, strict: bool = True)  $\rightarrow$  None

Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique.

 $\textbf{add\_listener}(\textit{listener: RecordUpdateListener, question: DNSQuestion} \mid \textit{list[DNSQuestion]} \mid \textit{None}) \rightarrow \\ \text{None}$ 

Adds a listener for a given question. The listener will have its update\_record method called when information is available to answer the question(s).

This function is threadsafe

 $remove\_listener(listener: RecordUpdateListener) \rightarrow None$ 

Removes a listener.

This function is threadsafe

 $\textbf{async\_add\_listener}. \ \textit{RecordUpdateListener}, \ \textit{question: DNSQuestion} \ | \ \textit{list[DNSQuestion]} \ | \ \textit{None}) \\ \rightarrow \text{None}$ 

Adds a listener for a given question. The listener will have its update\_record method called when information is available to answer the question(s).

This function is not threadsafe and must be called in the eventloop.

 $async\_remove\_listener(listener: RecordUpdateListener) \rightarrow None$ 

Removes a listener.

This function is not threadsafe and must be called in the eventloop.

```
send(out: DNSOutgoing, addr: str \mid None = None, port: int = 5353, v6 flow scope: tuple[()] \mid tuple[int, int]
            = (), transport: WrappedTransport | None = None) \rightarrow None
           Sends an outgoing packet threadsafe.
      async\_send(out: DNSOutgoing, addr: str \mid None = None, port: int = 5353, v6\_flow\_scope: tuple[()] \mid
                    tuple[int, int] = (), transport: \_WrappedTransport | None = None) \rightarrow None
           Sends an outgoing packet.
      close() \rightarrow None
           Ends the background threads, and prevent this instance from servicing further queries.
           This method is idempotent and irreversible.
      classmethod log_exception_debug(*logger data: Any) \rightarrow None
      classmethod log_exception_once(exc: Exception, *args: Any) \rightarrow None
      classmethod log_exception_warning(*logger\_data: Any) \rightarrow None
      classmethod log_warning_once(*args: Any) \rightarrow None
class AFL.automation.shared.ServerDiscovery.RunThread(coro)
      __init__(coro)
           This constructor should always be called with keyword arguments. Arguments are:
           group should be None; reserved for future extension when a ThreadGroup class is implemented.
           target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
           name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a
           small decimal number.
           args is a list or tuple of arguments for the target invocation. Defaults to ().
           kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.
           If a subclass overrides the constructor, it must make sure to invoke the base class constructor
           (Thread.__init__()) before doing anything else to the thread.
      run()
           Method representing the thread's activity.
           You may override this method in a subclass. The standard run() method invokes the callable object passed
           to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from
           the args and kwargs arguments, respectively.
class AFL.automation.shared.ServerDiscovery.ServerDiscovery
      ServerDiscovery class
      This class is used to discover AFL servers on the network using zeroconf requests that match a particular speci-
      fication.
      __init__()
      on_service_state_change(zeroconf: Zeroconf, service_type: str, name: str, state_change:
                                     ServiceStateChange) \rightarrow None
      async manage_service_info_to_list(zeroconf, service_type, name, append_or_remove)
      async get_service_info(zeroconf: Zeroconf, service\_type: str, name: str) \rightarrow None
```

#### async aio\_find\_server\_by\_name(service\_name)

#### discover\_server\_by\_name(service\_name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

#### async classmethod sa\_aio\_discover\_server\_by\_name(service name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

## classmethod sa\_discover\_server\_by\_name(service\_name)

Does a zeroconf request to find a named AFL-automation server on the network.

Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

### find\_server\_by\_name(service\_name)

Disambiguator for either matching or discovering a specific server by name

#### match\_server\_by\_name(service\_name)

Looks through the registry of discovered services for an exact name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

#### find\_server\_by\_partial\_name(service\_name)

Looks through the registry of discovered services for a partial name match. Returns a tuple of (host-name:port string, ServiceInfo object) if found, None if not found.

### find\_server\_by\_property\_match(property\_name, property\_value)

Looks through the registry of discovered services for a partial match in a property string. Returns a tuple of (hostname:port string, ServiceInfo object) if found, None if not found.

#### AFL.automation.shared.exceptions

#### **Exceptions**

| EmptyException         | Raised when a mixture runs out of mass or volume  |
|------------------------|---|
| MixingException        | Raised when a mixture cannot be made  |
| NoDeviceFoundException | Raised when no matching device can be found on the selected port                        |
| NotFoundError          |   |
| SerialCommsException   | Raised when the system receives a serial response it can't parse, likely a garbled line |

## AFL.automation.shared.exceptions.EmptyException

**exception** AFL.automation.shared.exceptions.**EmptyException** Raised when a mixture runs out of mass or volume

## AFL.automation.shared.exceptions.MixingException

exception AFL.automation.shared.exceptions.MixingException
 Raised when a mixture cannot be made

## AFL.automation.shared.exceptions.NoDeviceFoundException

**exception** AFL.automation.shared.exceptions.NoDeviceFoundException Raised when no matching device can be found on the selected port

### AFL.automation.shared.exceptions.NotFoundError

 $\textbf{exception} \ \, \textbf{AFL}. automation. shared. exceptions. \textbf{NotFoundError}$ 

## AFL.automation.shared.exceptions.SerialCommsException

**exception** AFL.automation.shared.exceptions.SerialCommsException

Raised when the system receives a serial response it can't parse, likely a garbled line

**exception** AFL.automation.shared.exceptions.**EmptyException**Raised when a mixture runs out of mass or volume

**exception** AFL.automation.shared.exceptions.MixingException Raised when a mixture cannot be made

**exception** AFL.automation.shared.exceptions.**SerialCommsException**Raised when the system receives a serial response it can't parse, likely a garbled line

**exception** AFL.automation.shared.exceptions.NoDeviceFoundException Raised when no matching device can be found on the selected port

exception AFL.automation.shared.exceptions.NotFoundError

#### AFL.automation.shared.serialization

#### **Functions**

```
deserialize(pickled_str)

is_serialized(obj)

serialize(obj)
```

## AFL.automation.shared.serialization.deserialize

AFL.automation.shared.serialization.deserialize(pickled\_str)

# AFL.automation.shared.serialization.is\_serialized

AFL.automation.shared.serialization.is\_serialized(obj)

#### AFL.automation.shared.serialization.serialize

AFL.automation.shared.serialization.serialize(obj)

# **Exceptions**

UnpicklingError

# AFL.automation.shared.serialization.UnpicklingError

exception AFL.automation.shared.serialization.UnpicklingError

AFL.automation.shared.serialization.serialize(obj)

AFL.automation.shared.serialization.is\_serialized(obj)

AFL.automation.shared.serialization.deserialize(pickled\_str)

## AFL.automation.shared.units

#### **Functions**

| <pre>enforce_units(value, unit_type)</pre> | Ensure that a number has units and convert to the default_units |
|--|---|
| <pre>get_unit_type(value)</pre>            |   |
| has_units(value)                           |   |
| is_concentration(value)                    |   |
| is_density(value)                          |   |
| is_mass(value)                             |   |
| is_molarity(value)                         |   |
| is_volume(value)                           |   |

## AFL.automation.shared.units.enforce\_units

AFL.automation.shared.units.enforce\_units(value, unit\_type)

Ensure that a number has units and convert to the default\_units

## AFL.automation.shared.units.get\_unit\_type

AFL.automation.shared.units.get\_unit\_type(value)

# AFL.automation.shared.units.has\_units

AFL.automation.shared.units.has\_units(value)

## AFL.automation.shared.units.is\_concentration

AFL.automation.shared.units.is\_concentration(value)

# AFL.automation.shared.units.is\_density

AFL.automation.shared.units.is\_density(value)

## AFL.automation.shared.units.is\_mass

AFL.automation.shared.units.is\_mass(value)

## AFL.automation.shared.units.is\_molarity

AFL.automation.shared.units.is\_molarity(value)

#### AFL.automation.shared.units.is volume

AFL.automation.shared.units.is\_volume(value)

AFL.automation.shared.units.has\_units(value)

AFL.automation.shared.units.is\_volume(value)

AFL.automation.shared.units.is\_molarity(value)

AFL.automation.shared.units.is\_mass(value)

AFL.automation.shared.units.is\_density(value)

AFL.automation.shared.units.is\_concentration(value)

AFL.automation.shared.units.get\_unit\_type(value)

AFL.automation.shared.units.enforce\_units(value, unit\_type)

Ensure that a number has units and convert to the default\_units

## AFL.automation.shared.utilities

#### **Functions**

```
has_units(value)

listify(obj)

makeRegistar()

mpl_plot_to_bytes([fig, format])

tprint(in_str)

xarray_to_bytes(ds[, format])
```

### AFL.automation.shared.utilities.has units

AFL.automation.shared.utilities.has\_units(value)

## AFL.automation.shared.utilities.listify

AFL.automation.shared.utilities.**listify**(obj)

# AFL.automation.shared.utilities.makeRegistar

AFL.automation.shared.utilities.makeRegistar()

## AFL.automation.shared.utilities.mpl\_plot\_to\_bytes

AFL.automation.shared.utilities.mpl\_plot\_to\_bytes(fig=None, format='svg')

## AFL.automation.shared.utilities.tprint

 ${\tt AFL.automation.shared.utilities.tprint} ({\it in\_str})$ 

## AFL.automation.shared.utilities.xarray\_to\_bytes

```
AFL.automation.shared.utilities.xarray_to_bytes(ds, format='svg')

AFL.automation.shared.utilities.listify(obj)

AFL.automation.shared.utilities.tprint(in_str)

AFL.automation.shared.utilities.makeRegistar()

AFL.automation.shared.utilities.mpl_plot_to_bytes(fig=None, format='svg')

AFL.automation.shared.utilities.xarray_to_bytes(ds, format='svg')
```

## AFL.automation.shared.widgetui

#### **Functions**

| <pre>clear_output([wait])</pre>                    | Clear the output of the current cell receiving output. |
|--|--|
| <pre>client_construct_ui(self[, return_ui,])</pre> |  |
| 1' 7 (% 1 · Γ · 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1    | D: 1 D d 1: (: 110 ( 1                                 |
| display(*objs[, include, exclude, metadata,])      | Display a Python object in all frontends.              |

## AFL.automation.shared.widgetui.clear output

AFL.automation.shared.widgetui.clear\_output(wait=False)

Clear the output of the current cell receiving output.

## **Parameters**

wait (bool [default: false]) - Wait to clear the output until new output is available to replace it.

## AFL.automation.shared.widgetui.client\_construct\_ui

AFL.automation.shared.widgetui.client\_construct\_ui(self, return\_ui=False, display\_ui=True)

#### AFL.automation.shared.widgetui.display

AFL.automation.shared.widgetui.display(\*objs, include=None, exclude=None, metadata=None, transient=None, display\_id=None, raw=False, clear=False, \*\*kwargs)

Display a Python object in all frontends.

By default all representations will be computed and sent to the frontends. Frontends can decide which representation is used and how.

In terminal IPython this will be similar to using print(), for use in richer frontends see Jupyter notebook examples with rich display logic.

#### **Parameters**

- \*objs (object) The Python objects to display.
- raw (bool, optional) Are the objects to be displayed already mimetype-keyed dicts of raw display data, or Python objects that need to be formatted before display? [default: False]
- **include** (*list*, *tuple or set*, *optional*) A list of format type strings (MIME types) to include in the format data dict. If this is set *only* the format types included in this list will be computed.
- **exclude** (*list*, *tuple* or *set*, *optional*) A list of format type strings (MIME types) to exclude in the format data dict. If this is set all format types will be computed, except for those included in this argument.
- metadata (dict, optional) A dictionary of metadata to associate with the output.
   mime-type keys in this dictionary will be associated with the individual representation formats, if they exist.
- **transient** (*dict*, *optional*) A dictionary of transient data to associate with the output. Data in this dict should not be persisted to files (e.g. notebooks).
- **display\_id**(*str*, *bool optional*) Set an id for the display. This id can be used for updating this display area later via update\_display. If given as *True*, generate a new *display\_id*
- **clear** (*bool*, *optional*) Should the output area be cleared before displaying anything? If True, this will wait for additional output before clearing. [default: False]
- \*\*kwargs (additional keyword-args, optional) Additional keyword-arguments are passed through to the display publisher.

#### Returns

**handle** – Returns a handle on updatable displays for use with update\_display(), if *display\_id* is given. Returns None if no *display\_id* is given (default).

## Return type

DisplayHandle

## **Examples**

```
>>> class Json(object):
...     def __init__(self, json):
...         self.json = json
...     def _repr_pretty_(self, pp, cycle):
...         import json
...         pp.text(json.dumps(self.json, indent=2))
...         def __repr__(self):
...         return str(self.json)
```

```
>>> d = Json({1:2, 3: {4:5}})
```

```
>>> print(d)
{1: 2, 3: {4: 5}}
```

```
>>> display(d)
{
   "1": 2,
```

(continues on next page)

(continued from previous page)

```
"3": {
    "4": 5
    }
}
```

```
>>> def int_formatter(integer, pp, cycle):
... pp.text('I'*integer)
```

```
>>> plain = get_ipython().display_formatter.formatters['text/plain']
>>> plain.for_type(int, int_formatter)
<function _repr_pprint at 0x...>
>>> display(7-5)
II
```

```
>>> del plain.type_printers[int]
>>> display(7-5)
2
```

```
 See also
update_display()
```

#### **Notes**

In Python, objects can declare their textual representation using the <u>\_\_repr\_\_</u> method. IPython expands on this idea and allows objects to declare other, rich representations including:

- HTML
- JSON
- PNG
- JPEG
- SVG
- LaTeX

A single object can declare some or all of these representations; all are handled by IPython's display system.

The main idea of the first approach is that you have to implement special display methods when you define your class, one for each representation you want to use. Here is a list of the names of the special methods and the values they must return:

- \_repr\_html\_: return raw HTML as a string, or a tuple (see below).
- \_repr\_json\_: return a JSONable dict, or a tuple (see below).
- \_repr\_jpeg\_: return raw JPEG data, or a tuple (see below).
- \_repr\_png\_: return raw PNG data, or a tuple (see below).
- \_repr\_svg\_: return raw SVG data as a string, or a tuple (see below).
- \_repr\_latex\_: return LaTeX commands in a string surrounded by "\$", or a tuple (see below).

# • \_repr\_mimebundle\_: return a full mimebundle containing the mapping

from all mimetypes to data. Use this for any mime-type not listed above.

The above functions may also return the object's metadata alonside the data. If the metadata is available, the functions will return a tuple containing the data and metadata, in that order. If there is no metadata available, then the functions will return the data only.

When you are directly writing your own classes, you can adapt them for display in IPython by following the above approach. But in practice, you often need to work with existing classes that you can't easily modify.

You can refer to the documentation on integrating with the display system in order to register custom formatters for already existing types (integrating\_rich\_display).

Added in version 5.4: display available without import

Added in version 6.1: display available without import

Since IPython 5.4 and 6.1 *display()* is automatically made available to the user without import. If you are using display in a document that might be used in a pure python context or with older version of IPython, use the following import at the top of your file:

from IPython.display import display

#### **Classes**

Markdown([data, url, filename, metadata])

#### AFL.automation.shared.widgetui.Markdown

Bases: TextDisplayObject

\_\_init\_\_(data=None, url=None, filename=None, metadata=None)

Create a display object given raw data.

When this object is returned by an expression or passed to the display function, it will result in the data being displayed in the frontend. The MIME type of the data should match the subclasses used, so the Png subclass should be used for 'image/png' data. If the data is a URL, the data will first be downloaded and then displayed.

#### **Parameters**

- data (unicode, str or bytes) The raw data or a URL or file to load the data from
- url (unicode) A URL to download the data from.
- **filename** (*unicode*) Path to a local file to load the data from.
- metadata (dict) Dict of metadata associated to be the object when displayed

## **Methods**

| init([data, url, filename, metadata]) | Create a display object given raw data. |
|---------------------------------------|---|
| reload()                              | Reload the raw data from file or URL.   |

#### **Attributes**

metadata

\_\_init\_\_(data=None, url=None, filename=None, metadata=None)

Create a display object given raw data.

When this object is returned by an expression or passed to the display function, it will result in the data being displayed in the frontend. The MIME type of the data should match the subclasses used, so the Png subclass should be used for 'image/png' data. If the data is a URL, the data will first be downloaded and then displayed.

#### **Parameters**

- data (unicode, str or bytes) The raw data or a URL or file to load the data from
- url (unicode) A URL to download the data from.
- **filename** (*unicode*) Path to a local file to load the data from.
- metadata (dict) Dict of metadata associated to be the object when displayed

metadata = None

#### reload()

Reload the raw data from file or URL.

AFL.automation.shared.widgetui.client\_construct\_ui(self, return\_ui=False, display\_ui=True)

**CHAPTER** 

# **SEVEN**

# MODULE DOCUMENTATION

AFL.automation. APIServer

AFL.automation.instrument

AFL.automation.loading

AFL.automation.prepare

AFL.automation.sample

AFL.automation.sample

# 7.1 AFL.automation

#### **Functions**

test() Run all tests using pytest.

# 7.1.1 AFL.automation.test

AFL.automation.test()

Run all tests using pytest.

AFL.automation.test()

Run all tests using pytest.

## **Modules**

APIServer

EpicsADLiveProcess

instrument

loading

prepare

sample\_env

shared

# 7.1.2 AFL.automation.EpicsADLiveProcess

#### **Modules**

AreaDetectorLive

Client

ReduceDaemon

# AFL.automation.EpicsADLiveProcess.AreaDetectorLive

# **Classes**

AreaDetectorLive([basepv, cam, filewriter, ...])

## AFL.automation.EpicsADLiveProcess.AreaDetectorLive.AreaDetectorLive

class AFL.automation.EpicsADLiveProcess.AreaDetectorLive.AreaDetectorLive(basepv='PIL5:', cam='cam1:', filewriter='TIFF1:', image='image1:')

```
Bases: object
__init__(basepv='PIL5:', cam='cam1:', filewriter='TIFF1:', image='image1:')
```

#### **Methods**

```
_init__([basepv, cam, filewriter, image])
      cbfunc([pvname, value, char_value])
      queuehandler()
      status()
     __init__(basepv='PIL5:', cam='cam1:', filewriter='TIFF1:', image='image1:')
     cbfunc(pvname=None, value=None, char_value=None, **kwargs)
     queuehandler()
     status()
class AFL.automation.EpicsADLiveProcess.AreaDetectorLive.AreaDetectorLive(basepv='PIL5:',
                                                                                     cam='cam1:',
                                                                                    filewriter='TIFF1:',
                                                                                     image='image1:')
     __init__(basepv='PIL5:', cam='cam1:', filewriter='TIFF1:', image='image1:')
     cbfunc(pvname=None, value=None, char_value=None, **kwargs)
     queuehandler()
     status()
```

## AFL.automation.EpicsADLiveProcess.Client

## **Classes**

```
Client([ip, port]) Communicate with NistoRoboto server on OT-2
```

# AFL.automation.EpicsADLiveProcess.Client.Client

```
class AFL.automation.EpicsADLiveProcess.Client.Client(ip='10.42.0.30', port='5000')
    Bases: object
    Communicate with NistoRoboto server on OT-2
    This class maps pipettor functions to HTTP REST requests that are sent to the NistoRoboto server
    __init__(ip='10.42.0.30', port='5000')
```

7.1. AFL.automation 821

#### **Methods**

```
_init__([ip, port])
       halt()
       home()
       load_instrument(name, mount, tip_rack_slots)
       load_labware(name, slot)
       logged_in()
       login(username)
       reset()
       set_queue_mode([debug_mode])
       transfer(source, dest, volume[, source_loc, ...])
                                                           Transfer fluid from one location to another
     __init__(ip='10.42.0.30', port='5000')
     logged_in()
     login(username)
     set_queue_mode(debug_mode=True)
     transfer(source, dest, volume, source_loc=None, dest_loc=None)
           Transfer fluid from one location to another
               Parameters
                   • source (str or list of str) - Source wells to transfer from. Wells should be spec-
                     ified as three character strings with the first character being the slot number.
                   • dest (str or list of str) - Destination wells to transfer from. Wells should be spec-
                     ified as three character strings with the first character being the slot number.
                   • volume (float) – volume of fluid to transfer in microliters
     load_labware(name, slot)
     load_instrument(name, mount, tip_rack_slots)
     reset()
     home()
     halt()
class AFL.automation.EpicsADLiveProcess.Client.Client(ip='10.42.0.30', port='5000')
     Communicate with NistoRoboto server on OT-2
     This class maps pipettor functions to HTTP REST requests that are sent to the NistoRoboto server
```

```
__init__(ip='10.42.0.30', port='5000')
logged_in()
login(username)
set_queue_mode(debug_mode=True)
transfer(source, dest, volume, source_loc=None, dest_loc=None)
    Transfer fluid from one location to another
    Parameters
```

- **source** (*str or list of str*) Source wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **dest** (*str or list of str*) Destination wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.
- **volume** (*float*) volume of fluid to transfer in microliters

```
load_labware(name, slot)
load_instrument(name, mount, tip_rack_slots)
reset()
home()
halt()
```

#### AFL.automation.EpicsADLiveProcess.ReduceDaemon

#### **Classes**

```
ReduceDaemon(app, reduction_queue, ...[, ...])
```

#### AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon

```
Bases: Thread
__init__(app, reduction_queue, integrator, results, mask=None, npts=500)
This constructor should always be called with keyword arguments. Arguments are:
group should be None; reserved for future extension when a ThreadGroup class is implemented.
target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.
```

args is a list or tuple of arguments for the target invocation. Defaults to ().

7.1. AFL.automation 823

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

#### **Methods**

| init(app, reduction_queue, integrator,) | This constructor should always be called with keyword arguments. |
|---|--|
| <pre>getName()</pre>                    | Return a string used for identification purposes only.           |
| isDaemon()                              | Return whether this thread is a daemon.                          |
| is_alive()                              | Return whether the thread is alive.                              |
| <pre>join([timeout])</pre>              | Wait until the thread terminates.                                |
| run()                                   | Method representing the thread's activity.                       |
| setDaemon(daemonic)                     | Set whether this thread is a daemon.                             |
| setName(name)                           | Set the name string for this thread.                             |
| start()                                 | Start the thread's activity.                                     |
| terminate()                             |  |
|   |  |

#### **Attributes**

| daemon    | A boolean value indicating whether this thread is a daemon thread.            |
|-----------|---|
| ident     | Thread identifier of this thread or None if it has not been started.          |
| name      | A string used for identification purposes only.                               |
| native_id | Native integral thread ID of this thread, or None if it has not been started. |

**\_\_init\_\_**(app, reduction\_queue, integrator, results, mask=None, npts=500)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

## terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

## property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

7.1. AFL.automation 825

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

## start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

## class AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon(app, reduction\_queue,

integrator, results,
mask=None, npts=500)

\_\_init\_\_(app, reduction\_queue, integrator, results, mask=None, npts=500)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.\_\_init\_\_()) before doing anything else to the thread.

## terminate()

#### run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

## **CHAPTER**

## **EIGHT**

# **INDICES AND TABLES**

- genindex
- modindex
- search

## **PYTHON MODULE INDEX**

| a  | 44  |
|--|---|
| AFL.automation, 15                                   | AFL.automation.loading.PressureController,45                          |
| AFL.automation.APIServer, 16                         | AFL.automation.loading.PressureControllerAsPump,                      |
| AFL.automation.APIServer.APIServer, 17               | 46  |
| AFL.automation.APIServer.Client, 19                  | AFL.automation.loading.PushPullSelectorSampleCell,                    |
| AFL.automation.APIServer.Driver, 21                  | 47  |
| AFL.automation.APIServer.DummyDriver, 23             | ${\tt AFL.automation.loading.RSoXSSolutionSampleCell},$               |
| AFL.automation.APIServer.DummyOT2Driver, 24          | 49  |
| AFL.automation.APIServer.LoggerFilter, 25            | AFL.automation.loading.SainSmartRelay,50                              |
| AFL.automation.APIServer.QueueDaemon, 25             | AFL.automation.loading.SampleCell,52                                  |
| AFL.automation.EpicsADLiveProcess, 26                | AFL.automation.loading.Sensor, 52                                     |
| AFL.automation.EpicsADLiveProcess.AreaDetect         | orAfiveautomation.loading.SensorCallbackThread,                       |
| 26   | 33  |
| AFL.automation.EpicsADLiveProcess.Client, 26         | ${\tt AFL.automation.loading.SensorPollingThread},$                   |
| AFL.automation.EpicsADLiveProcess.ReduceDaem         | on, 56  |
| 27   | AFL.automation.loading.SerialDevice, 57                               |
| AFL.automation.instrument, 28                        | AFL.automation.loading.SyringePump, 57                                |
| AFL.automation.instrument.DummySAS, 28               | AFL.automation.loading.Tubing, 58                                     |
| AFL.automation.instrument.FileCamera, 29             | ${\tt AFL. automation. loading. Two Selector Blowout Sample Cell},\\$ |
| AFL.automation.instrument.I22SAXS, 29                | 58  |
| AFL.automation.instrument.NetworkCamera, 29          | AFL.automation.loading.UltimusVPressureController,                    |
| AFL.automation.instrument.SeabreezeUVVis, 30         | 60  |
| AFL.automation.instrument.SpecScreen_Driver,         | AFL.automation.loading.ViciMultiposSelector,                          |
| 31   | 61  |
| AFL.automation.loading, 32                           | AFL.automation.prepare, 61  |
| AFL.automation.loading.CetoniMultiPosValve,          | AFL.automation.prepare.Component, 63                                  |
| 34   | AFL.automation.prepare.DeckBuilderWidget, 64                          |
| AFL.automation.loading.ChemyxSyringePump, 34         | AFL.automation.prepare.Dummy_OT2_Driver, 66                           |
| AFL.automation.loading.DigitalOutPressureCon         | tropleautomation.prepare.factory, 74                                  |
| 36   | AFL.automation.prepare.OT2Client, 67                                  |
| AFL.automation.loading.DoubleViciMultiposSel         | ector, automation.prepare.Preparewidget, 68                           |
| 37   | AFL.automation.prepare.PrepType, 68                                   |
| AFL.automation.loading.DummyPump,38                  | AFL.automation.prepare.SampleSeriesWidget,70                          |
| AFL.automation.loading.FlowSelector,38               | AFL.automation.prepare.StockBuilderWidget,71                          |
| AFL.automation.loading.LoadStopperDriver, 39         | AFL.automation.prepare.SweepBuilderWidget,72                          |
| ${\tt AFL.automation.loading.MultiChannelRelay}, 40$ | AFL.automation.prepare.utilities, 74                                  |
| ${\tt AFL.automation.loading.NE1kSyringePump}, 40$   | AFL.automation.sample, 89   |
| AFL.automation.loading.OneSelectorBlowoutSam         | pledelautomation.sample.CastingServer,89                              |
| $\Delta 1$   | AFL.automation.sample_env,/5  |
| AFL.automation.loading.PneumaticPressureSamp         | leAELlautomation.sample_env.TemperatureDeck,75                        |
| 42   | AFL.automation.shared, 76   |
| AFL.automation.loading.PneumaticSampleCell,          | AFL.automation.shared.DataLabelerWidget,76                            |

```
AFL.automation.shared.DatasetWidget, 78
AFL.automation.shared.DiffractionLabeler, 80
AFL.automation.shared.exceptions, 86
AFL.automation.shared.MutableQueue, 82
AFL.automation.shared.PersistentConfig, 83
AFL.automation.shared.serialization, 86
AFL.automation.shared.ServerDiscovery, 84
AFL.automation.shared.units, 87
AFL.automation.shared.utilities, 87
AFL.automation.shared.widgetui, 88
```

830 Python Module Index

## **INDEX**

| Symbols  | method), 528  |
|--|---|
| bytes() (AFL.automation.instrument.SeabreezeUVVis <del>.P</del> dftl-                          | () (AFL.automation.prepare.PrepareWidget.Layout         |
| method), 213   | method), 539  |
| call() (AFL.automation.APIServer.APIServer.Flaskdel  | () (AFL.automation.prepare.PrepareWidget.Text           |
| method) 134  | method), 551  |
| memod), 194<br>contains() (AFL.automation.APIServer.APIServer.IP <del>Velsion</del>            | () (AFL.automation.prepare.PrepareWidget.VBox           |
| class method), 135   | method), 559  |
| contains() (AFL.automation.prepare.Component.Prepflype_  | () (AFL.automation.prepare.SampleSeriesWidget.Button    |
| class method) 307  | method), 569  |
| contains() (AFL.automation.prepare.PrepType.Enum_del   | () (AFL.automation.prepare.SampleSeriesWidget.Checkbox  |
| class mathod) 180  | meinoa), 5/0  |
| contains() (AFL.automation.prepare.PrepType.Prep <del>Type</del> 1                             | () (AFL.automation.prepare.SampleSeriesWidget.FloatText |
| class mathod) 400  | meinoa), 567  |
| contains() (AFL.automation.shared.ServerDiscover <del>y.H<sup>o</sup>versi</del>               | (AFL.automation.prepare.SampleSeriesWidget.HBox         |
| class method) 788  | memoa), 393   |
| contains() (AFL.automation.shared.ServerDiscover <del>y.S</del> ervice                         | State Hange omation.prepare.SampleSeriesWidget.IntText  |
| class method) 802  | <i>method</i> ), 603                                    |
| del() (AFL.automation.prepare.DeckBuilderWidget.B <del>utl</del> de <sup>1</sup>               | () (AFL.automation.prepare.SampleSeriesWidget.Label     |
| method) 402  | <i>metnoa</i> ), 611                                    |
| del() (AFL.automation.prepare.DeckBuilderWidget.C <del>he</del> ckDox                          | () (AFL.automation.prepare.SampleSeriesWidget.Layout    |
| mathod) 410  | meinoa), 022  |
| memod), 410<br>del() (AFL.automation.prepare.DeckBuilderWidget.Dropadown                       | (() (AFL.automation.prepare.SampleSeriesWidget.Text     |
| method) 423  | meinoa), 032  |
| del() (AFL.automation.prepare.DeckBuilderWidget.H <del>B</del> ol <sup>el</sup>                | (() (AFL.automation.prepare.SampleSeriesWidget.VBox     |
| method) 431  | <i>methoa</i> ), 640                                    |
| del() (AFL.automation.prepare.DeckBuilderWidget.L <del>ab</del> elel                           | (() (AFL.automation.prepare.SweepBuilderWidget.Button   |
| method) 439  | metnoa), 033  |
| memod), 439<br>del() (AFL.automation.prepare.DeckBuilderWidget.L <del>ay</del> out             | (() (AFL.automation.prepare.SweepBuilderWidget.Checkbox |
| mathad) 150  | meinoa), 001  |
| memod), 450<br>del() (AFL.automation.prepare.DeckBuilderWidget.T <del>ext</del> <sup>del</sup> | () (AFL.automation.prepare.SweepBuilderWidget.HBox      |
| method) 457  | meinoa), 009  |
| del() (AFL.automation.prepare.DeckBuilderWidget.V <del>Bo</del> x <sup>d</sup> el              | () (AFL.automation.prepare.SweepBuilderWidget.Label     |
| method) 465  | metnoa), 0//  |
| del() (AFL.automation.prepare.PrepareWidget.Buttondel  | (AFL.automation.prepare.SweepBuilaerWiaget.Layout       |
| method) 495  | metnoa), 088  |
| del() (AFL.automation.prepare.PrepareWidget.Check <del>bo</del> x <sup>del</sup>               | (AFL.automation.prepare.SweepButtaerwiaget.1ext         |
| mathod) 502  | meinoa), 09/  |
| del() (AFL.automation.prepare.PrepareWidget.Dropd <del>ow</del> n =                            | (Ar L. automation. prepare. Sweep Buttaer wtaget. v Box |
| method), 512   | meinoa), 703  |
| ucl() (II Liamonianompreparen repare vitageni12ex  | (AFL.automation.prepare.Solution method),               |
| method), 520   | 391 (AFI automation prepare factory Solution            |
| del() (AFL.automation.prepare.PrepareWidget.Labeleq(   | (AFL.automation.prepare.factory.Solution                |

| method), 715  | method), 162, 163   |
|---|---|
| getitem() (AFL.automation.APIServer.APIServer.IPVerinint_           | () (AFL.automation.APIServer.Driver.Driver  |
| class method), 135  | method), 21, 166, 167, 171  |
| getitem() (AFL.automation.APIServer.Driver.Persistentistaints       | g_() (AFL.automation.APIServer.Driver.PersistentConfig  |
| method), 170  | method), 169, 170   |
| getitem() (AFL.automation.prepare.Component.PrepTypmit_             | () (AFL.automation.APIServer.DummyDriver.Driver   |
| class method), 397  | method), 173, 174   |
| getitem() (AFL.automation.prepare.PrepType.Enuminit_                | () (AFL.automation.APIServer.DummyDriver.DummyDriver  |
| class method), 489  | method), 23, 176, 178, 179  |
| getitem() (AFL.automation.prepare.PrepType.PrepTypainit_            | () (AFL.automation.APIServer.DummyOT2Driver.Driver  |
| class method), 490  | method), 181  |
| getitem() (AFL.automation.shared.PersistentConfig.Perisisten        | <u>tConfi</u> gFL.automation.APIServer.DummyOT2Driver.DummyDriver   |
| method), 83, 774, 776   | method), 24, 183, 184, 186  |
| getitem() (AFL.automation.shared.ServerDiscovery.IP\\arista         | n_() (AFL.automation.APIServer.LoggerFilter.LoggerFilter  |
| class method), 788  | method), 25, 186, 187   |
| getitem() (AFL.automation.shared.ServerDiscovery.SerimatS           | taleXlAdFilgeautomation.APIServer.QueueDaemon.DataTrashcan  |
| class method), 802  | method), 187, 189   |
| hash() (AFL.automation.prepare.Component.Componeninit_              | () (AFL.automation.APIServer.QueueDaemon.QueueDaemon  |
| method), 64, 396, 398   | method), 25, 189, 190, 192  |
| hash() (AFL.automation.prepare.Solute method),init_                 | $\_$ () (AFL.automation. Epics ADLive Process. Area Detector Live. Area   |
| 389   | method), 26, 820, 821   |
|   | _() (AFL.automation.EpicsADLiveProcess.Client.Client  |
| method), 391  | method), 26, 821, 822   |
| hash() (AFL.automation.prepare.Solvent method),init_                | $\_$ () (AFL.automation.EpicsADLiveProcess.ReduceDaemon.Re |
| 393   | method), 27, 823, 824, 826  |
| hash() (AFL.automation.prepare.factory.Solutioninit_                |   |
| method), 714  | method), 193, 194   |
| init() (AFL.automation.APIServer.APIServer.APISer <u>ve</u> init_   |   |
| method), 18, 97, 99, 157  | method), 28, 196–198  |
| init() (AFL.automation.APIServer.APIServer.CORSinit_                |   |
| method), 102, 103   | method), 29, 198, 199   |
| init() (AFL.automation.APIServer.APIServer.FileHandlenit_           |   |
| method), 103, 104   | method), 199, 200   |
| init() (AFL.automation.APIServer.APIServer.Flaskinit_               |   |
| method), 107, 113   | method), 29, 202–204  |
| init() (AFL.automation.APIServer.APIServer.IPVersioninit_           |   |
| method), 135  | method), 30, 205  |
| init() (AFL.automation.APIServer.APIServer.JWTManageirt_            |   |
| method), 136, 137   | method), 206  |
| init() (AFL.automation.APIServer.APIServer.Logger <u>Fil</u> terit_ |   |
| method), 141  | method), 208  |
| init() (AFL.automation.APIServer.APIServer.MutableQurine_           |   |
| method), 141  | method), 209  |
| init() (AFL.automation.APIServer.APIServer.QueueDaimint_            |   |
| method), 142, 143   | method), 30, 215, 216, 218  |
| init() (AFL.automation.APIServer.APIServer.SMTPHainMet_             | •   |
| method), 145, 146   | method), 219, 220   |
| init() (AFL.automation.APIServer.APIServer.ServiceInfinit_          | · · ·   |
| method), 148, 149   | method), 31, 222, 224   |
| init() (AFL.automation.APIServer.APIServer.Zeroconfinit_            | ·   |
| method), 152, 154   | method), 34, 226, 227   |
| init() (AFL.automation.APIServer.Client.Clientinit_                 |   |
| method), 20, 159, 161, 164  | method), 227  |

\_\_init\_\_() (AFL.automation.APIServer.Client.ServerDisc<u>ov</u>inyit\_\_() (AFL.automation.loading.ChemyxSyringePump.ChemyxConn

```
method), 36, 228, 229, 233
                                                                method), 43, 284, 286, 290
__init__() (AFL.automation.loading.ChemyxSyringePump.GlnertyxSy) in Penantomation.loading.PneumaticPressureSampleCell.Sam
        method), 35, 230-232
                                                                method), 288
__init__() (AFL.automation.loading.ChemyxSyringePump.SymintgePumpAFL.automation.loading.PneumaticPressureSampleCell.defa
        method), 232
                                                                method), 288, 289
__init__() (AFL.automation.loading.DigitalOutPressureController.D)giAHDuutPressuiveCloadiohler.PneumaticSampleCell.Driver
                                                                method), 292
        method), 37, 234-236
__init__() (AFL.automation.loading.DigitalOutPressureCoritmalter.R)@(Adrik.Gattmalteon.loading.PneumaticSampleCell.PneumaticSa
        method), 236
                                                                method), 44, 294, 296, 300
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Selivnixt,DA</u>)b(AVIL:MadbipasSalkaading.PneumaticSampleCell.SampleCell
        method), 37, 237, 239
                                                                method), 298
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Seliviotr.Fl6</u>)vS&leEtantomation.loading.PneumaticSampleCell.defaultdict
        method), 238
                                                                method), 298, 299
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Selinivt.Vi¢iMAlfipanSoluctioon.</u>loading.PressureController.PressureControl
        method), 238, 239
                                                                method), 301
__init__() (AFL.automation.loading.DummyPump.DummyPump_() (AFL.automation.loading.PressureControllerAsPump.Pressur
        method), 38, 240–242
                                                                method), 46, 303, 305
__init__() (AFL.automation.loading.DummyPump.SerialDeviriet__() (AFL.automation.loading.PressureControllerAsPump.SerialD
        method), 241, 242
                                                                method), 304
__init__() (AFL.automation.loading.DummyPump.SyringePimixt__() (AFL.automation.loading.PressureControllerAsPump.Syringe
        method), 242
                                                                method), 304
__init__() (AFL.automation.loading.FlowSelector.FlowSeleitairt__() (AFL.automation.loading.PushPullSelectorSampleCell.Driver
                                                                method), 306, 307
        method), 243
__init__() (AFL.automation.loading.LoadStopperDriver.Clizmit__() (AFL.automation.loading.PushPullSelectorSampleCell.PushF
        method), 244, 245
                                                                method), 47, 309, 311, 316
__init__() (AFL.automation.loading.LoadStopperDriver.Drinit__() (AFL.automation.loading.PushPullSelectorSampleCell.Sampl
        method), 247, 248
                                                                method), 313
__init__() (AFL.automation.loading.LoadStopperDriver.LoadStopp@DtAFdr.automation.loading.PushPullSelectorSampleCell.Tubing
        method), 39, 249, 251, 261
                                                                method), 313, 314
__init__() (AFL.automation.loading.LoadStopperDriver.<u>SerismiRoll</u>ing(AhFdadutomation.loading.PushPullSelectorSampleCell.defaul
        method), 252, 253
                                                                method), 314, 315
__init__() (AFL.automation.loading.LoadStopperDriver.StophindCDylAFL.automation.loading.RSoXSSolutionSampleCell.Driver
        method), 255, 256
                                                                method), 318
__init__() (AFL.automation.loading.LoadStopperDriver.<u>StophirtadCB</u>yQAFL.automation.loading.RSoXSSolutionSampleCell.RSoXSS
        method), 259
                                                                method), 49, 320, 322, 327
__init__() (AFL.automation.loading.MultiChannelRelay.MultiChann@lRelayautomation.loading.RSoXSSolutionSampleCell.SampleC
        method), 262
                                                                method), 324
__init__() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePuAlfL.automation.loading.RSoXSSolutionSampleCell.Tubing
        method), 40, 263, 264, 266
                                                                method), 324, 325
__init__() (AFL.automation.loading.NE1kSyringePump.SeriadDevice) (AFL.automation.loading.RSoXSSolutionSampleCell.defaultd.
        method), 265
                                                                method), 325, 326
__init__() (AFL.automation.loading.NE1kSyringePump.SyringePum(i) (AFL.automation.loading.SainSmartRelay.MultiChannelRelay
        method), 265
                                                                method), 329
__init__() (AFL.automation.loading.OneSelectorBlowoutSainpiltCeNDtAFdr.automation.loading.SainSmartRelay.SainSmartRelay
        method), 268
                                                                method), 51, 330, 331
__init__() (AFL.automation.loading.OneSelectorBlowout<u>SairpiltCeNQAASelautonBltiorouloSalinpl</u>eGaethSmartRelay.SerialDevice
         method), 42, 270, 272, 280
                                                                method), 331
__init__() (AFL.automation.loading.OneSelectorBlowout<u>Sa</u>inpilt<u>CeNJVAASelautonRitivonUtSainpileSainpileCell</u>.SampleCell
        method), 274, 276
                                                                method), 333
__init__() (AFL.automation.loading.OneSelectorBlowout<u>Sa</u>inpileCellAleAdIldiatomation.loading.Sensor.DummySensor1
                                                                method), 52, 333, 334, 339
        method), 279
__init__() (AFL.automation.loading.PneumaticPressureSamipticCell(DriAFL.automation.loading.Sensor.DummySensor2
        method), 281, 282
                                                                method), 53, 336, 337, 340
__init__() (AFL.automation.loading.PneumaticPressureSaniplicCell(BneumAfidPausomasampledinlg.Sensor.Sensor
```

|         | method), 52, 339  | method), 65, 419, 471  |
|---------|---|--|
| init_   |   | e <b>rConAfilnination</b> ation.prepare.DeckBuilderWidget.DeckBuilderWi  |
|         | method), 56, 341, 355   | method), 65, 420, 471  |
| init    |   | rCdUAdkThutenuttion.prepare.DeckBuilderWidget.DeckBuilderWi  |
|         | method), 54, 341, 342, 353  | method), 65, 420, 471  |
| init    |   | Landar, 65, 126, 171 Landar, 65, 17 |
|         | method), 55, 344, 345, 355  | method), 421, 423  |
| ini+    | () (AFL.automation.loading.SensorCallbackThre <u>ad<b>iStop</b>L</u>        |  |
|         |   | method), 430, 431  |
| : m: ±  | method), 55, 347, 348, 354  |  |
| 1n1t_   | _() (AFL.automation.loading.SensorCallbackThreadiStiopL                     |  |
|         | method), 55, 350, 351, 354  | method), 437, 439  |
| 1n1t_   | _() (AFL.automation.loading.SensorPollingThread.Sinion)                     |  |
|         | method), 56, 356, 357, 359  | method), 445, 450  |
| init_   | _() (AFL.automation.loading.SerialDevice.Serial <u>Dev<b>ina</b></u> t_     |  |
|         | method), 57, 360  | method), 455, 457  |
| init_   | _() (AFL.automation.loading.SyringePump.Syring <u>eP<b>imi</b></u> pt_      |  |
|         | method), 361  | method), 464, 465  |
| init_   | _() (AFL.automation.loading.Tubing.Tubinginit_                              | _() (AFL.automation.prepare.Dummy_OT2_Driver.Driver  |
|         | method), 58, 362  | method), 472, 473  |
| init_   | () (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpil&Q             | <u>CeM.Drivelr.automation.prepare.Dummy_OT2_Driver.Dummy_OT2</u>   |
|         | method), 363, 364   | method), 66, 475, 476, 478   |
| init_   | _() (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpiltC            | LeU.Samp(ACE)llautomation.prepare.MassBalance  |
|         | method), 366  | method), 384, 385  |
| init    | _() (AFL.automation.loading.TwoSelectorBlowout <u>Sa<b>inpil</b>tC</u>      |  |
|         | method), 366  | method), 480, 481  |
| init    | () (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpile(             |  |
|         | method), 59, 367, 369, 373  | method), 483, 485  |
| init    | () (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpleC              |  |
|         | method), 371, 372   | method), 486, 487  |
| ini+    | () (AFL.automation.loading.UltimusVPressureCont <b>ivoli</b> ty,            |  |
| 1111 (_ | () (AF L. automation. todating. Ottimus v1 ressure Comming_<br>method), 375 | method), 386   |
| 22.     |   |  |
| 1n1t_   | _() (AFL.automation.loading.UltimusVPressureContinular_                     |  |
|         | method), 60, 376, 377   | method), 489   |
| 1n1t_   | _() (AFL.automation.loading.ViciMultiposSelector.FlaviSe                    |  |
|         | method), 378  | method), 489   |
| init_   | _() (AFL.automation.loading.ViciMultiposSelecto <u>r.S<b>àriùt</b>D</u>     |  |
|         | method), 378  | method), 490, 491  |
| init_   | _() (AFL.automation.loading.ViciMultiposSelector <u>.V<b>i</b>ciM</u> u     |  |
|         | method), 61, 379, 380   | method), 492, 494  |
| init_   | () (AFL.automation.prepare.Component.Compon <u>en</u> init_                 | _() (AFL.automation.prepare.PrepareWidget.Checkbox   |
|         | method), 64, 395, 396, 398  | method), 500, 502  |
| init_   | () (AFL.automation.prepare.Component.PrepTyp <u>e</u> _init_                | _() (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget  |
|         | method), 397  | method), 508, 509  |
| init_   | _() (AFL.automation.prepare.ComponentDBinit_                                | _() (AFL.automation.prepare.PrepareWidget.Dropdown   |
|         | method), 382  | method), 510, 512  |
| init    |   | _() (AFL.automation.prepare.PrepareWidget.HBox   |
|         | 382, 383  | method), 519, 520  |
| init    | () (AFL.automation.prepare.DeckBuilderWidget. <u>Bu<b>imi</b></u> t_        |  |
|         | method), 400, 402   | method), 526, 528  |
| ini+    |   |  |
| 1n1t_   | () (AFL.automation.prepare.DeckBuilderWidget. <u>Ch</u> irokbo.             |  |
| 2002.   | method), 408, 410   | method), 534, 539  |
| 1n1t_   | · · · · · · · · · · · · · · · · · · ·                                       | _() (AFL.automation.prepare.PrepareWidget.PrepareWidget  |
|         | method), 416, 418   | method), 69, 544, 565  |
| init_   | () (AFL.automation.prepare.DeckBuilderWidget. <u>De<b>i:hR.t</b>il</u>      | ld&nWAAgktautomation.prepare.PrepareWidget.PrepareWidget_Mod   |

|         | method), 545   | method), 72, 649, 650   |
|---------|--|---|
| init_   | _() (AFL.automation.prepare.PrepareWidget.Prep <u>ar</u> <b>aWid</b> ge                    |   |
|         | method), 545   | method), 651, 653   |
| init_   |  | <u>W</u> (dgeAFL.automation.prepare.SweepBuilderWidget.Checkbox           |
|         | method), 545, 546  | method), 659, 661   |
| init    | _() (AFL.automation.prepare.PrepareWidget.Stock <u>Bnithde</u> r)                          |   |
|         | method), 547, 548  | method), 667, 669   |
| init    | _() (AFL.automation.prepare.PrepareWidget.SweepBirikher                                    |   |
|         | method), 548   | method), 675, 677   |
| init    | () (AFL.automation.prepare.PrepareWidget.Textinit_   |   |
|         | method), 549, 551  | method), 683, 688   |
| init    |  | () (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder                |
|         | method), 558, 559  | method), 73, 693, 711   |
| init    |  | () (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder                |
|         | 387  | method), 73, 694, 711   |
| init_   |  | () (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder                |
|         | method), 388   | method), 73, 694, 695, 711  |
| ini+    | () (AFL.automation.prepare.SampleSeriesWidget <u>.Bimort_</u>                              |   |
| 1111 (_ | method), 566, 568  | method), 695, 697   |
| ini+    | () (AFL.automation.prepare.SampleSeriesWidget <u>.Chrisbo</u>                              |   |
| 1111    | method), 574, 576  | method), 704, 705   |
| ini+    | () (AFL.automation.prepare.SampleSeriesWidget <u>.Cli<b>v</b>it</u> t_                     |   |
| 1111 (_ | method), 582, 584  | method), 713, 714   |
| ini+    | () (AFL.automation.prepare.SampleSeriesWidget <u>_Fli<b>vaiT</b>e</u> ;                    |   |
| 1111 (_ | () (AT L. automation. prepare. Sample Series Wiaget <u>, Final Re</u><br>method), 585, 587 | method), 716  |
| ::+     |  |   |
| 1n1t_   | () (AFL.automation.prepare.SampleSeriesWidget_HBnit_                                       |   |
| 22.     | method), 594, 595  | method), 89, 718, 720, 730  |
| 1n1t_   | () (AFL.automation.prepare.SampleSeriesWidget <u>_In</u> iTeixt_                           |   |
| 22.     | method), 601, 603  | method), 721, 723   |
| 1n1t_   | () (AFL.automation.prepare.SampleSeriesWidget_Laiheit_                                     |   |
| 22.     | method), 609, 611  | method), 724, 725   |
| 1n1t_   | _() (AFL.automation.prepare.SampleSeriesWidget_Layaint_                                    |   |
|         | method), 617, 622  | method), 727, 729   |
| 1n1t_   | _() (AFL.automation.prepare.SampleSeriesWidget <u>.Sa</u> mpleS                            |   |
| 22.     | method), 70, 627, 628, 646   | method), 731, 732   |
| 1n1t_   |  | Sefie(WFdgatutModtibn.sample_env.TemperatureDeck.TemperatureL             |
|         | method), 71, 629, 646  | method), 75, 734–736  |
| init_   |  | Sefie (Widgetut Vinution.shared.DataLabeler Widget.DataLabeler Mod        |
|         | method), 71, 629, 630, 647   | method), 77, 738, 748   |
| 1n1t_   |  | _() (AFL.automation.shared.DataLabelerWidget.DataLabelerVie               |
|         | method), 630, 632  | method), 77, 738, 748   |
| init_   |  | () (AFL.automation.shared.DataLabelerWidget.DataLabelerWid                |
|         | method), 639, 640  | method), 77, 739, 740, 747  |
| init_   |  | $\_()$ (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder            |
|         | 388, 389   | method), 744, 745   |
| init_   |  | _() (AFL.automation.shared.DatasetWidget.DatasetWidget                    |
|         | method), 390, 391  | method), 78, 749, 750, 756  |
| init_   |  | $\_()$ (AFL.automation.shared.DatasetWidget.DatasetWidget $\_$ Mode       |
|         | 392, 393   | method), 79, 752, 757   |
| init_   |  | ld() WAdJetautomation.shared.DatasetWidget.DatasetWidget_View             |
|         | method), 71, 647–649   | method), 80, 753, 754, 758  |
| init_   | () (AFL.automation.prepare.StockBuilderWidget. <u>Sto<b>ickBu</b>i</u>                     | ld <b>@)WAdde.</b> tau <b>Mordat</b> ion.shared.DatasetWidget.defaultdict |

*method*), 755

\_\_init\_\_() (AFL.automation.prepare.StockBuilderWidget.<u>StockBuild@) WAd Jeta.Wiomation.shared.DiffractionLabeler.DiffractionLab</u>

method), 72, 649, 650

|         | method), 81, 759, 760, 768   | class method), 802  |
|---------|--|---|
| init_   | _() (AFL.automation.shared.DiffractionLabeler.Diffr <b>heti</b> on     | LdkAFIMndomation.APIServer.APIServer.IPVersion  |
|         | method), 81, 761, 768  | class method), 135  |
| init_   | _() (AFL.automation.shared.DiffractionLabeler.Diffraction              | LabelFiViertomation.prepare.Component.PrepType  |
|         | method), 81, 761, 769  | class method), 397  |
| init    | _() (AFL.automation.shared.DiffractionLabeler.Ordi <b>hat</b> End      |   |
|         | method), 765, 766  | class method), 489  |
| init    | () (AFL.automation.shared.MutableQueue.Muta <u>ble<b>Qu</b>uu</u>      |   |
| 1111    | method), 82, 769–771   | class method), 490  |
| init    |  |   |
| 1111    | _() (AFL.automation.shared.PersistentConfig.Mutable <b>M</b> app       |   |
|         | method), 771   | class method), 788  |
| 1n1t_   | • •  | ### African Server Discovery. Service State Change  |
|         | method), 83, 773–775   | class method), 802  |
| init_   |  | Panow(ArL.automation.APIServer.Driver.PersistentConfig  |
|         | method), 777, 778  | method), 170  |
| init_   | _() (AFL.automation.shared.ServerDiscovery.Asy <u>nc<b>Sæ</b>vi</u> ce | $Penf_O$ () (AFL. automation. shared. Persistent Config. Persistent Configure () (AFL. automation. shared. Persistent Configure () () (AFL. automation. shared. Persistent Configure () () () (AFL. automation. shared. Persistent Configure () () () () () () () () () () () () () |
|         | method), 780, 781  | method), 83, 774, 776   |
| init_   | _() (AFL.automation.shared.ServerDiscovery.Asy <u>ncZet<b>v</b>co</u>  | n(f) (AFL.automation.instrument.SeabreezeUVVis.Path   |
|         | method), 784, 785  | method), 213  |
| init_   | _() (AFL.automation.shared.ServerDiscovery.AsyncZeroco                 | nfServiceTypes  |
|         | method), 786, 787  |   |
| init    | _() (AFL.automation.shared.ServerDiscovery.IPV@\\\e\r                  | (AFL.automation.APIServer.APIServer.Flask   |
|         | method), 788   |   |
| init    | _() (AFL.automation.shared.ServerDiscovery.Run <b>Three</b> der        | attribute), 113   |
|         | method), 84, 788, 789, 808   |   |
| ini+    |  | attribute), 111   |
| 1111 (_ | _() (AFL.automation.shared.ServerDiscovery.ServænBingon                |   |
|         | method), 85, 791, 792, 808   | method), 211  |
| 1n1 t_  | _() (AFL.automation.shared.ServerDiscovery.ServiceRrove                |   |
|         | method), 793, 794  | method), 104  |
| 1n1t_   | _() (AFL.automation.shared.ServerDiscovery.ServiceInfore               |   |
|         | method), 797, 799  | method), 146  |
| init_   | _() (AFL.automation.shared.ServerDiscovery.ServiceStryteA              | HERR tomation.prepare.ComponentDB method),  |
|         | method), 802   | 382   |
| init_   | _() (AFL.automation.shared.ServerDiscovery.Zeroant_arr                 | ay()(AFL.automation.APIServer.QueueDaemon.DataTrashcan  |
|         | method), 803, 805  | method). 188  |
| init_   | _() (AFL.automation.shared.widgetui.Markdown add_cat                   | ch() (AFL.automation.prepare.Deck method),  |
|         | method), 817, 818  | 384   |
| iter_   | _() (AFL.automation.APIServer.APIServer.IPVersized_cla                 | SS() (AFL automation prepare DeckRuilderWidget Rutton   |
|         | class method), 135   | method), 403  |
| iter    |  | ss() (AFL.automation.prepare.DeckBuilderWidget.Checkbox   |
|         | method), 64, 396, 398  | method), 410  |
| iter    |  | ss() (AFL.automation.prepare.DeckBuilderWidget.Dropdown   |
| 1       | class method), 397   |   |
| itor    |  | method), 423  |
| 1       | · · · · · · · · · · · · · · · · · · ·                                  | ss() (AFL.automation.prepare.DeckBuilderWidget.HBox   |
|         | class method), 489   | method), 431  |
| iter_   | _() (AFL.automation.prepare.PrepType.PrepType add_cla                  |   |
|         | class method), 490   | method), 439  |
| iter_   | $\_()$ (AFL.automation.prepare.Solute method), add_cla                 | ss() (AFL.automation.prepare.DeckBuilderWidget.Text   |
|         | 389  | method), 457  |
| iter_   | _() (AFL.automation.prepare.Solvent method), add_cla                   | ss() (AFL.automation.prepare.DeckBuilderWidget.VBox   |
|         | 393  | method) 465   |
| iter_   | _() (AFL.automation.shared.ServerDiscovery.IPVerdip_ncla               | ss() (AFL.automation.prepare.PrepareWidget.Button   |
|         | class method), 788   | method), 495  |
| iter    | _() (AFL.automation.shared.ServerDiscovery.ServiceState(               |   |

```
add_class() (AFL.automation.prepare.PrepareWidget.Chaddopipette()
                                                                                                                   (AFL.automation.prepare.Deck
             method), 502
                                                                                              method), 384
add_class() (AFL.automation.prepare.PrepareWidget.Draphbyprep_targets() (AFL.automation.prepare.Dummy OT2 Driver.Du
                                                                                              method), 66, 477, 479
             method), 512
add_class() (AFL.automation.prepare.PrepareWidget.HBadd_sample() (AFL.automation.prepare.SampleSeries
             method), 520
                                                                                              method), 388
add_class() (AFL.automation.prepare.PrepareWidget.Labadd_service() (AFL.automation.shared.ServerDiscovery.AsyncZeroconfl
             method), 528
                                                                                              method), 787
add_class() (AFL.automation.prepare.PrepareWidget.Textadd_service_listener()
                                                                                              (AFL.automation.APIServer.APIServer.Zeroconf
             method), 551
add_class() (AFL.automation.prepare.PrepareWidget.VBox
                                                                                              method), 155
                                                                                add_service_listener()
             method), 559
add_class() (AFL.automation.prepare.SampleSeriesWidget.Button (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                              method), 806
             method), 569
add_class() (AFL.automation.prepare.SampleSeriesWidgetddhectdandard_routes()
             method), 576
                                                                                              (AFL.automation.APIServer.APIServer.APIServer
add_class() (AFL.automation.prepare.SampleSeriesWidget.FloatTextethod), 18, 99, 157
             method), 587
                                                                                add_stock() (AFL.automation.prepare.Deck method),
add_class() (AFL.automation.prepare.SampleSeriesWidget.HBox 384
             method), 595
                                                                                add_stock()
                                                                                                        (AFL.automation.prepare.MassBalance
add_class() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 385
                                                                                add_stocks_to_deck()
             method), 603
add_class() (AFL.automation.prepare.SampleSeriesWidget.Label (AFL.automation.prepare.PrepareWidget.StockBuilderWidget
                                                                                              method), 548
             method), 611
method), 632
                                                                                              (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidge
add_class() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 71, 648, 650
                                                                                add_stocks_to_deck()
             method), 640
add_class() (AFL.automation.prepare.SweepBuilderWidget.Button (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderWidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.StockBuilderwidget.Sto
                                                                                              method), 72, 649, 650
             method), 654
add_class() (AFL.automation.prepare.SweepBuilderWidgatlChreklget() (AFL.automation.prepare.Deck method),
             method), 661
                                                                                              384
add_class() (AFL.automation.prepare.SweepBuilderWidgadblBcemplate_filter()
                                                                                              (AFL.automation.APIServer.APIServer.Flask
             method), 669
add_class() (AFL.automation.prepare.SweepBuilderWidget.Label method), 120
                                                                                add_template_global()
             method), 677
add_class() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                              (AFL.automation.APIServer.APIServer.Flask
             method), 697
                                                                                              method), 121
add_class()(AFL.automation.prepare.SweepBuilderWidgatd\Bommplate_test()
             method), 705
                                                                                              (AFL.automation.APIServer.APIServer.Flask
add_component_from_name()
                                                                                              method), 121
             (AFL.automation.prepare.factory.Solution
                                                                                add_traits() (AFL.automation.prepare.DeckBuilderWidget.Button
             method), 715
                                                                                              method), 403
add_component_from_name()
                                                                                add_traits() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
             (AFL. automation. prepare. Solution
                                                                 method),
                                                                                              method), 410
             391
                                                                                add_traits() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
add_container()
                                   (AFL.automation.prepare.Deck
                                                                                              method), 423
                                                                                add_traits()(AFL.automation.prepare.DeckBuilderWidget.HBox
             method), 384
add_interactive() (AFL.automation.prepare.ComponentDB
                                                                                              method), 431
             method), 382
                                                                                add\_traits() (AFL.automation.prepare.DeckBuilderWidget.Label
add_listener()(AFL.automation.APIServer.APIServer.Zeroconf method), 439
                                                                                add_traits()(AFL.automation.prepare.DeckBuilderWidget.Layout
             method), 156
add_listener() (AFL.automation.shared.ServerDiscovery.Zeroconfinethod), 450
             method), 807
                                                                                add_traits() (AFL.automation.prepare.DeckBuilderWidget.Text
```

```
method), 458
                                                                                                              add_url_rule() (AFL.automation.APIServer.APIServer.Flask
add_traits()(AFL.automation.prepare.DeckBuilderWidget.VBox method), 119
                                                                                                              add_vertical_line()
                  method), 465
                                                                                                                                 (AFL.automation.shared.DataLabelerWidget.DataLabelerView
add_traits() (AFL.automation.prepare.PrepareWidget.Button
                  method), 495
                                                                                                                                 method), 77, 739, 748
add_traits()(AFL.automation.prepare.PrepareWidget.Chaddbwertical_line()
                                                                                                                                 (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerVi
                  method), 503
add_traits()(AFL.automation.prepare.PrepareWidget.Dropdown method), 81, 762, 769
                  method), 512
                                                                                                              Added (AFL.automation.shared.ServerDiscovery.ServiceStateChange
add\_traits() (AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                 attribute), 802
                  method), 520
                                                                                                              addFilter() (AFL. automation. APIServer. APIServer. FileHandler
add_traits() (AFL.automation.prepare.PrepareWidget.Label
                                                                                                                                 method), 104
                  method), 528
                                                                                                              addFilter() (AFL.automation.APIServer.APIServer.SMTPHandler
add_traits() (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                 method), 146
                                                                                                              additional_claims_loader()
                  method), 539
add_traits() (AFL.automation.prepare.PrepareWidget.Text
                                                                                                                                 (AFL.automation.APIServer.APIServer.JWTManager
                  method), 552
                                                                                                                                 method), 138
add_traits()(AFL.automation.prepare.PrepareWidget.VMadditional_headers_loader()
                                                                                                                                 (AFL.automation.APIServer.APIServer.JWTManager
                  method), 559
add_traits() (AFL.automation.prepare.SampleSeriesWidget.Buttonmethod), 138
                  method), 569
                                                                                                              addMode() (AFL.automation.loading.ChemyxSyringePump.ChemyxConnec
add_traits() (AFL.automation.prepare.SampleSeriesWidget.Checkbroathod), 36, 230, 234
                                                                                                              addresses (AFL. automation. API Server. API Server. Service Info
                  method), 577
add_traits() (AFL.automation.prepare.SampleSeriesWidget.FloatTaxtribute), 149
                  method), 588
                                                                                                              addresses (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
add_traits() (AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 781
                  method), 595
                                                                                                              addresses (AFL. automation. shared. Server Discovery. Service Info
add_traits()(AFL.automation.prepare.SampleSeriesWidget.IntTextattribute), 799
                                                                                                              addresses_by_version()
                  method), 603
\verb|add_traits()| (AFL. automation. prepare. Sample Series Widget. Label (AFL. automation. API Server. API Server. Service Information. API Server. AP
                  method), 611
                                                                                                                                 method), 150
add_traits()(AFL.automation.prepare.SampleSeriesWidgeddhesses_by_version()
                                                                                                                                 (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                  method), 622
add_traits()(AFL.automation.prepare.SampleSeriesWidget.Text method), 782
                  method), 633
                                                                                                              addresses_by_version()
add_traits() (AFL.automation.prepare.SampleSeriesWidget.VBox (AFL.automation.shared.ServerDiscovery.ServiceInfo
                  method), 640
                                                                                                                                method), 800
add_traits() (AFL.automation.prepare.SweepBuilderWidgetdE@y.6AFL.automation.loading.ChemyxSyringePump.ChemyxConnection
                  method), 654
                                                                                                                                 method), 36, 230, 234
add_traits()(AFL.automation.prepare.SweepBuilderWidaetjGstccpmotocol_order()
                                                                                                                                 (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
                  method), 661
add_traits() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 71, 629, 646
                                                                                                              advanceSample() (AFL.automation.loading.PneumaticPressureSampleCe
                  method), 669
add_traits() (AFL.automation.prepare.SweepBuilderWidget.Label method), 43, 286, 291
                  method), 677
                                                                                                              advertise_zeroconf()
add_traits() (AFL.automation.prepare.SweepBuilderWidget.Layou(AFL.automation.APIServer.APIServer.APIServer
                  method), 688
                                                                                                                                 method), 18, 99, 157
add_traits()(AFL.automation.prepare.SweepBuilderWidkFL.Taxttomation
                  method), 698
                                                                                                                       module, 15, 819
add_traits() (AFL.automation.prepare.SweepBuilderWidkFLVBoxtomation.APIServer
                                                                                                                       module, 16, 90
                  method), 705
add_unqueued_routes()
                                                                                                              AFL.automation.APIServer.APIServer
                  (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                       module, 17, 91
                  method), 18, 99, 158
                                                                                                              AFL.automation.APIServer.Client
```

| module, 19, 159                                       | module, 41, 267                                     |
|---|---|
| AFL.automation.APIServer.Driver                       | AFL.automation.loading.PneumaticPressureSampleCell  |
| module, 21, 165                                       | module, 42, 281                                     |
| AFL.automation.APIServer.DummyDriver                  | AFL.automation.loading.PneumaticSampleCell          |
| module, 23, 173                                       | module, 44, 291                                     |
| AFL.automation.APIServer.DummyOT2Driver               | AFL.automation.loading.PressureController           |
| module, 24, 180                                       | module, 45, 301                                     |
| AFL.automation.APIServer.LoggerFilter                 | AFL.automation.loading.PressureControllerAsPump     |
| module, 25, 186                                       | module, 46, 302                                     |
| AFL.automation.APIServer.QueueDaemon                  | AFL.automation.loading.PushPullSelectorSampleCell   |
| module, 25, 187                                       | module, 47, 305                                     |
| AFL.automation.EpicsADLiveProcess                     |   |
|   | AFL.automation.loading.RSoXSSolutionSampleCell      |
| module, 26, 820                                       | module, 49, 317                                     |
| AFL.automation.EpicsADLiveProcess.AreaDetecto         |   |
| module, 26, 820                                       | module, 50, 328                                     |
| AFL.automation.EpicsADLiveProcess.Client              | AFL.automation.loading.SampleCell                   |
| module, 26, 821                                       | module, 52, 332                                     |
| ${\tt AFL.automation.EpicsADLiveProcess.ReduceDaemo}$ |   |
| module, 27, 823                                       | module, 52, 333                                     |
| AFL.automation.instrument                             | AFL.automation.loading.SensorCallbackThread         |
| module, 28, 193                                       | module, 53, 341                                     |
| AFL.automation.instrument.DummySAS                    | AFL.automation.loading.SensorPollingThread          |
| module, 28, 193                                       | module, 56, 355                                     |
| AFL.automation.instrument.FileCamera                  | AFL.automation.loading.SerialDevice                 |
| module, 29, 198                                       | module, 57, 359                                     |
| AFL.automation.instrument.I22SAXS                     | AFL.automation.loading.SyringePump                  |
| module, 29, 199                                       | module, 57, 360                                     |
| AFL.automation.instrument.NetworkCamera               | AFL.automation.loading.Tubing                       |
| module, 29, 204                                       | module, 58, 362                                     |
| AFL.automation.instrument.SeabreezeUVVis              | AFL.automation.loading.TwoSelectorBlowoutSampleCell |
| module, 30, 205                                       | module, 58, 363                                     |
| AFL.automation.instrument.SpecScreen_Driver           | AFL.automation.loading.UltimusVPressureController   |
| module, 31, 219                                       | module, 60, 374                                     |
| AFL.automation.loading                                | AFL.automation.loading.ViciMultiposSelector         |
| module, 32, 224                                       | module, 61, 377                                     |
| AFL.automation.loading.CetoniMultiPosValve            | AFL.automation.prepare                              |
| module, 34, 226                                       | module, 61, 380                                     |
| AFL.automation.loading.ChemyxSyringePump              | AFL.automation.prepare.Component                    |
| module, 34, 228                                       | module, 63, 394                                     |
| AFL.automation.loading.DigitalOutPressureCont         |   |
| 5 5   |   |
| module, 36, 234                                       | module, 64, 399                                     |
| AFL.automation.loading.DoubleViciMultiposSele         |   |
| module, 37, 237                                       | module, 66, 472                                     |
| AFL.automation.loading.DummyPump                      | AFL.automation.prepare.factory                      |
| module, 38, 240                                       | module, 74, 712                                     |
| AFL.automation.loading.FlowSelector                   | AFL.automation.prepare.OT2Client                    |
| module, 38, 243                                       | module, 67, 480                                     |
| AFL.automation.loading.LoadStopperDriver              | AFL.automation.prepare.PrepareWidget                |
| module, 39, 244                                       | module, 68, 491                                     |
| AFL.automation.loading.MultiChannelRelay              | AFL.automation.prepare.PrepType                     |
| module, 40, 262                                       | module, 68, 488                                     |
| AFL.automation.loading.NE1kSyringePump                | AFL.automation.prepare.SampleSeriesWidget           |
| module, 40, 263                                       | module, 70, 565                                     |
| AFL. automation. loading. One Selector Blowout Samp   | LANGE Lautomation.prepare.StockBuilderWidget        |

module, 71, 647

attribute), 620

| AFL.automation.prepare.SweepBuilderWidget            | attribute), 686  |
|--|--|
| module, 72, 650                                      | $\verb align_items   (AFL. automation. prepare. Deck Builder Widget. Layout$   |
| AFL.automation.prepare.utilities                     | attribute), 448  |
| module, 74, 716                                      | align_items(AFL.automation.prepare.PrepareWidget.Layout  |
| AFL.automation.sample                                | attribute), 537  |
| module, 89, 717                                      | align_items(AFL.automation.prepare.SampleSeriesWidget.Layout   |
| AFL.automation.sample.CastingServer                  | attribute), 620  |
| module, 89, 717                                      | align_items(AFL.automation.prepare.SweepBuilderWidget.Layout   |
| AFL.automation.sample_env                            | attribute), 686  |
| module, 75, 731                                      | align_self(AFL.automation.prepare.DeckBuilderWidget.Layout   |
| AFL.automation.sample_env.TemperatureDeck            | attribute), 448  |
| module, 75, 731                                      | align_self(AFL.automation.prepare.PrepareWidget.Layout   |
| AFL.automation.shared                                | attribute), 537  |
| module, 76, 736                                      | align_self(AFL.automation.prepare.SampleSeriesWidget.Layout  |
| AFL.automation.shared.DataLabelerWidget              | attribute), 620  |
| module, 76, 737                                      | align_self(AFL.automation.prepare.SweepBuilderWidget.Layout  |
| AFL.automation.shared.DatasetWidget                  | attribute), 686  |
| module, 78, 748                                      | alive() (AFL.automation.loading.LoadStopperDriver.SensorPollingThre  |
| AFL.automation.shared.DiffractionLabeler             | method), 254   |
| module, 80, 759                                      | alive() (AFL.automation.loading.SensorPollingThread.SensorPollingTh  |
| AFL.automation.shared.exceptions                     | method), 57, 357, 359  |
| module, 86, 809                                      | All (AFL.automation.APIServer.APIServer.IPVersion at-  |
| AFL.automation.shared.MutableQueue                   | tribute), 135  |
| module, 82, 769                                      | All (AFL.automation.shared.ServerDiscovery.IPVersion   |
| AFL.automation.shared.PersistentConfig               | attribute), 788  |
| module, 83, 771                                      | analyze_stocks_cb()  |
| AFL.automation.shared.serialization                  | (AFL.automation.prepare.PrepareWidget.StockBuilderWidget   |
| module, 86, 810                                      | method), 548   |
| AFL.automation.shared.ServerDiscovery                | analyze_stocks_cb()  |
| module, 84, 776                                      | (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidg  |
| AFL.automation.shared.units                          | method), 71, 648, 649  |
| module, 87, 811                                      | anchor (AFL.automation.instrument.SeabreezeUVVis.Path  |
| AFL.automation.shared.utilities                      | property), 213   |
| module, 87, 813                                      | APIServer (class in AFL.automation.APIServer.APIServer),   |
| AFL.automation.shared.widgetui                       | 18, 97, 157  |
| module, 88, 814                                      | app (AFL.automation.loading.LoadStopperDriver.LoadStopperDriver  |
| after_request() (AFL.automation.APIServer.APIServer  |  |
| method), 126   | app (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelector   |
|  |  |
| after_request_funcs                                  | property), 272   |
| (AFL.automation.APIServer.APIServer.Flask            | app (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelector   |
| attribute), 131                                      | property), 277   |
| aio_find_server_by_name()                            | app (AFL.automation.loading.PneumaticPressureSampleCell.PneumaticP   |
| (AFL.automation.APIServer.Client.ServerDiscove       |  |
| method), 163   | app (AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCe  |
| aio_find_server_by_name()                            | property), 45, 296, 300  |
|  | Depart (AF)L.automation.loading.PushPullSelectorSampleCell.PushPullSelec   |
| method), 85, 792, 808                                | property), 48, 311, 317  |
|  | dappl(Aydutuutomation.loading.RSoXSSolutionSampleCell.RSoXSSolutionS   |
| attribute), 448                                      | property), 50, 322, 327  |
| align_content (AFL.automation.prepare.PrepareWidget. | [app (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowo |
| auriniue 1 13 /                                      | DEODERNI 19 JUS J /4   |

 $\verb|align_content| (AFL. automation. prepare. Sweep Builder Widget. Layout$ 

840 Index

 $\verb|align_content| (AFL. automation. prepare. Sample Series Widget. League \texttt{x} \texttt{t}() (AFL. automation. API Server. API Server. Flask) + (AFL. automation. API Server. API Server. Flask) + (AFL. automation. API Server. API$ 

method), 133

```
app_ctx_globals_class
                                                                                                method), 807
             (AFL.automation.APIServer.APIServer.Flask
                                                                                  async_clear_cache()
             attribute), 111
                                                                                                (AFL.automation.APIServer.APIServer.ServiceInfo
apply_isel() (AFL.automation.shared.DatasetWidget.DatasetWidgetnethod), 150
             method), 79, 751, 757
                                                                                  async_clear_cache()
apply_isel() (AFL.automation.shared.DatasetWidget.DatasetWidgetMadatomation.shared.ServerDiscovery.AsyncServiceInfo
             method), 80, 753, 758
                                                                                               method), 782
apply_protocol_order()
                                                                                  async_clear_cache()
             (AFL.automation.prepare.PrepareWidget.SampleSeriesWidgetFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 546
                                                                                               method), 800
apply_protocol_order()
                                                                                  async\_close() (AFL. automation. shared. Server Discovery. Async Zeroconf
             (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWeitlget), 786
             method), 70, 628, 646
                                                                                  apply_protocol_order_cb()
                                                                                                class method), 787
             (AFL.automation.prepare.PrepareWidget.SampleSassian&Vidget_service_info()
             method), 546
                                                                                                (AFL.automation.APIServer.APIServer.Zeroconf
apply_protocol_order_cb()
                                                                                                method), 155
             (AFL.automation.prepare.SampleSeriesWidget.SampleSeriestWidgetVice_info()
             method), 70, 628, 646
                                                                                                (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
{\tt apply\_sel()} \ (AFL. automation. shared. Dataset Widget. Dataset Widget method), 786
             method), 79, 751, 757
                                                                                  async_get_service_info()
apply_sel() (AFL.automation.shared.DatasetWidget.DatasetWidget(AFLautomation.shared.ServerDiscovery.Zeroconf
             method), 80, 753, 758
                                                                                                method), 806
AreaDetectorLive
                                                (class
                                                                            in async_notify_all() (AFL.automation.APIServer.APIServer.Zeroconf
             AFL.automation.EpicsADLiveProcess.AreaDetectorLive), method), 154
             26, 820, 821
                                                                                  async_notify_all() (AFL.automation.shared.ServerDiscovery.Zeroconf
args (AFL. automation. prepare. Component. Parse Exception
                                                                                                method), 805
                                                                                  async_register_service()
             attribute), 398
as_posix() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                               (AFL.automation.APIServer.APIServer.Zeroconf
             method), 213
                                                                                                method), 155
as_uri()(AFL.automation.instrument.SeabreezeUVVis.Pathsync_register_service()
                                                                                                (AFL. automation. shared. Server Discovery. A sync Zero conf
             method), 213
aspirate_rate()(AFL.automation.prepare.OT2Client.OT2Client method), 785
             method), 67, 485, 487
                                                                                  async_register_service()
aspirate_rate() (AFL.automation.sample.CastingServer.OT2ClientAFL.automation.shared.ServerDiscovery.Zeroconf
             method), 729
                                                                                                method), 806
assign_targets() (AFL.automation.sample.CastingServeaceSymptimpeSinower_all_service_listeners()
             method), 89, 720, 730
                                                                                                (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
async_add_listener()
                                                                                                method), 786
             (AFL.automation.APIServer.APIServer.Zeroconf async_remove_listener()
             method), 157
                                                                                                (AFL.automation.APIServer.APIServer.Zeroconf
async_add_listener()
                                                                                                method), 157
             (AFL.automation.shared.ServerDiscovery.Zeroconfsync_remove_listener()
             method), 807
                                                                                                (AFL. automation. shared. Server Discovery. Zero conf
async_add_service_listener()
                                                                                               method), 807
             (AFL.automation.shared.ServerDiscovery.AsyncZersynnof_remove_service_listener()
             method), 786
                                                                                                (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
async_cancel() (AFL.automation.shared.ServerDiscovery.AsyncSermathBd) w 386
             method), 779
                                                                                  async_request() (AFL.automation.APIServer.APIServer.ServiceInfo
async_check_service()
                                                                                                method), 150
             (AFL. automation. APIS erver. APIS erver. Zeroconf \ {\tt async\_request()}\ (AFL. automation. shared. Server Discovery. Async Service (AFL. automation. Shared. Server Discovery). As not server (AFL. automation. Shared. Server (AFL. automation. Shared. Server (AFL. automation. Shared. Server (AFL. automation. Shared. Shared.
                                                                                               method), 782
             method), 156
async_check_service()
                                                                                  async_request() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                               method), 800
```

| async_send() (AFL.automation.APIServer.APIServer.Zer<br>method), 157   | oaconnfic_u                    | pdate_service()<br>(AFL.automation.APIServer.APIServer.Zero | conf                  |
|--|--------------------------------|---|-----------------------|
| async_send() (AFL.automation.shared.ServerDiscovery.2  | 7eroconf                       |   | cong                  |
| method), 808   |                                | pdate_service()   |                       |
| async_to_sync() (AFL.automation.APIServer.APIServer.   | -                              | (AFL.automation.shared.ServerDiscovery.As                   | sync7eroconf          |
| method), 124   | .r iusk                        | method), 786  | synczeroconj          |
| async_unregister_all_services()  | acume II                       | pdate_service()   |                       |
|  |                                | _   |                       |
| (AFL.automation.APIServer.APIServer.Zeroconf   |                                | (AFL.automation.shared.ServerDiscovery.Ze                   | eroconj               |
| method), 156   |                                | method), 806  | u Camia a Lufa        |
| async_unregister_all_services() (AFL.automation.shared.ServerDiscovery.Async2  | Zeroconf                       |   |                       |
| method), 785   | async_w                        | <pre>ait() (AFL.automation.APIServer.APIServe</pre>         | r.Zeroconf            |
| <pre>async_unregister_all_services()</pre>   |                                | method), 154  |                       |
| (AFL. automation. shared. Server Discovery. Zeroco   | <i>n</i> gsync_w               | <pre>ait() (AFL.automation.shared.ServerDiscov</pre>        | very.AsyncServiceInfo |
| method), 807   |                                | method), 782  |                       |
| <pre>async_unregister_service()</pre>  | async_w                        | <pre>ait() (AFL.automation.shared.ServerDiscov</pre>        | very.ServiceInfo      |
| (AFL.automation.APIServer.APIServer.Zeroconf   |                                | method), 800  |                       |
| method), 156   | async_w                        | ait()(AFL.automation.shared.ServerDiscov                    | very.Zeroconf         |
| <pre>async_unregister_service()</pre>  |                                | method), 805  |                       |
| (AFL.automation.shared.ServerDiscovery.AsyncZ  | Z <i>avsy</i> zno <u>f</u> w   | ait_for_start()   |                       |
| method), 786   | - 0                            | (AFL.automation.APIServer.APIServer.Zero                    | conf                  |
| async_unregister_service()   |                                | method), 154  | ,                     |
| (AFL.automation.shared.ServerDiscovery.Zeroco  | nafsvnc w                      |   |                       |
| method), 807   | <i>y</i> - <i>y</i> - <u>–</u> | (AFL.automation.shared.ServerDiscovery.Ze                   | eroconf               |
| async_update_records()   |                                | method), 805  |                       |
| (AFL.automation.APIServer.APIServer.ServiceIn  | f <b>A</b> syncSe              |   | in                    |
| method), 150   | wisyncoc                       | AFL.automation.shared.ServerDiscovery),                     | · · ·                 |
| async_update_records()   |                                | 777   |                       |
| (AFL.automation.shared.ServerDiscovery.AsyncS  | A STANSTON                     |   | in                    |
| method), 779   | ars yataba                     | AFL.automation.shared.ServerDiscovery),                     | ııı                   |
| async_update_records()   |                                | 780   |                       |
| (AFL.automation.shared.ServerDiscovery.AsyncS  | Marindah                       |   | in                    |
| method), 782   | из упылы                       | AFL.automation.shared.ServerDiscovery),                     | iii                   |
| async_update_records()   |                                | 784   |                       |
|  | . Parma7a                      | ,   | :                     |
| (AFL:automation.shared.ServerDiscovery.Service   | е <b>н</b> Бушк <i>и</i> ге    |   | in                    |
| method), 795   |                                | AFL.automation.shared.ServerDiscovery),                     |                       |
| async_update_records()   | I-6 - ( 1                      | 786   | 0                     |
| (AFL.automation.shared.ServerDiscovery.Service   |                                |   | 0                     |
| method), 800   | auto_f1                        | nd_instance_path()  | 1                     |
| <pre>async_update_records_complete()</pre>   | 0                              | (AFL.automation.APIServer.APIServer.Flash                   | K                     |
| (AFL.automation.APIServer.APIServer.ServiceIn  | fo                             | method), 116  |                       |
| method), 150   | D                              |   |                       |
| <pre>async_update_records_complete()</pre>   | В                              |   |                       |
| (AFL.automation.shared.ServerDiscovery.AsyncS  | waiaake                        | Wiff&\$s() (AFL.automation.prepare.MassBald                 | ance                  |
| method), 779   |                                | method), 386  |                       |
| <pre>async_update_records_complete()</pre>   | balance                        | r (AFL.automation.prepare.Sample proper                     | ty),                  |
| (AFL. automation. shared. Server Discovery. Async States and States are also becomes a support of the states and states are also becomes a support of the stat | ServiceInfo                    | 2387  |                       |
| method), 782   | BaseCom                        | ponent (AFL.automation.prepare.Componen                     | t.PrepType            |
| <pre>async_update_records_complete()</pre>   |                                | attribute), 397   |                       |
| (AFL.automation.shared.ServerDiscovery.Service   | Beseccin                       | ponent (AFL.automation.prepare.PrepTvpe.l                   | PrepType              |
| method), 795   |                                | attribute), 68, 490, 491                                    | 1 71                  |
| <pre>async_update_records_complete()</pre>   | BaseMix                        | ture (AFL.automation.prepare.Component.F                    | PrepType              |
| (AFL.automation.shared.ServerDiscovery.Service method), 800  | eInfo                          | attribute), 397   | r vr                  |

| BaseMixture (AFL.automation.prepare.PrepType.PrepTypattribute), 68, 490, 491   | pe (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSe<br>method), 42, 272, 281   |
|--|--|
| <pre>before_first_request()</pre>  | blowOutCellLegacy()  |
| (AFL.automation.APIServer.APIServer.Flask method), 121   | (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSe method), 277  |
| before_first_request_funcs   | blowOutCellLegacy()  |
| (AFL.automation.APIServer.APIServer.Flask attribute), 114  | (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolut<br>method), 50, 323, 328  |
| before_request() (AFL.automation.APIServer.API   | <i>eh</i> Fhw@utCellLegacy()   |
| method), 126   | (AFL. automation. loading. Two Selector Blow out Sample Cell. Two Selectors and the selection of the selec |
| before_request_funcs   | method), 60, 370, 374  |
| (AFL.automation.APIServer.APIServer.Flask attribute), 131  | blueprints (AFL.automation.APIServer.APIServer.Flask attribute), 114   |
| blockUntilStatusStopped()  | blur() (AFL.automation.prepare.DeckBuilderWidget.Button  |
| (AFL.automation.loading.ChemyxSyringePump.C  |  |
| method), 35, 232, 233  | blur() (AFL.automation.prepare.DeckBuilderWidget.Checkbox  |
| blockUntilStatusStopped()  | method), 410   |
|  | n <b>bbble (DiAifdl.Outbrassure pæptro</b> dlDeckBuilderWidget.Dropdown  |
| method), 235   | method), 423   |
| blockUntilStatusStopped()  | blur() (AFL.automation.prepare.DeckBuilderWidget.HBox  |
| (AFL.automation.loading.DigitalOutPressureCo   |  |
| method), 236   | blur() (AFL.automation.prepare.DeckBuilderWidget.Label   |
| blockUntilStatusStopped()  | method), 439   |
|  | Pholopr() (AFL.automation.prepare.DeckBuilderWidget.Text   |
| method), 38, 241, 243  | method), 458   |
| blockUntilStatusStopped()  | blur() (AFL.automation.prepare.DeckBuilderWidget.VBox  |
| (AFL.automation.loading.NE1kSyringePump.NE   |  |
| method), 41, 265, 267  | blur() (AFL.automation.prepare.PrepareWidget.Button  |
| blockUntilStatusStopped()  | method), 495   |
| (AFL. automation. loading. Pressure Controller. P   | es <b>olvæ@ontAtelle.n</b> utomation.prepare.PrepareWidget.Checkbox  |
| method), 46, 302   | method), 503   |
| <pre>blockUntilStatusStopped()</pre>   | blur() (AFL.automation.prepare.PrepareWidget.Dropdown  |
| (AFL. automation. loading. Pressure Controller As Full Formula (AFL) and the property of the   | Pump.Press <b>met</b> God)r,dH&rAsPump   |
| method), 47, 303, 305  | blur() (AFL.automation.prepare.PrepareWidget.HBox  |
| <pre>blockUntilStatusStopped()</pre>   | method), 520   |
| $(AFL. automation. loading. {\it Ultimus VP ressure Constitution}) \\$   | tr <b>bl/ar.P)</b> es <b>(AdeCautrollæt</b> ion.prepare.PrepareWidget.Label  |
| method), 375   | method), 528   |
| <pre>blockUntilStatusStopped()</pre>   | blur() (AFL.automation.prepare.PrepareWidget.Text  |
| $(AFL. automation. loading. {\it Ultimus VP ressure Constitution}) and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution}. The {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution}. The {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution}. The {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution}. The {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution} and {\it Ultimus VP ressure Constitution}. The {\it Ultimus VP ressure Constitution} and {\it Ultimus $ |  |
| method), 376   | blur() (AFL.automation.prepare.PrepareWidget.VBox  |
| $\verb blowOutCell()  (AFL. automation. loading. One Selector Blow OutCell())  (AFL. automation. loading. loading. Outcell())  (AFL. automation. loading. Outcell())  (AFL. automation. loading. Outcell())  (AFL. automation. loading. Ou$   | *  |
| method), 42, 272, 281  | $\verb blur()  (AFL. automation. prepare. Sample Series Widget. Button$  |
| ${\tt blowOutCell()} \ (AFL. automation. loading. One Selector Blow OutCell()) \ (AFL. automation. loading. One Selector Blow OutCell$   |  |
| method), 277   | blur() (AFL.automation.prepare.SampleSeriesWidget.Checkbox   |
| $\verb blowOutCell()  (AFL. automation. loading. PushPullSelect$   |  |
| method), 48, 313, 317  | blur() (AFL.automation.prepare.SampleSeriesWidget.FloatText  |
| $\verb blowOutCell()  (AFL. automation. loading. RSoXSS olution   Constraints   Constrain$   |  |
| method), 50, 323, 328  | blur() (AFL.automation.prepare.SampleSeriesWidget.HBox   |
| $\verb blowOutCell()  (AFL. automation. loading. Two Selector Blow OutCell())  (AFL. automation. loading. loading. Loading. Loading. Republic Blow OutCell())  (AFL. automation. loading. Re$   |  |
| method), 60, 370, 374  | blur() (AFL.automation.prepare.SampleSeriesWidget.IntText  |
| blowOutCellForcedAir()   | method), 603   |
|  | olla Calt. Puch Pulla elevatati Sample Gall Sample Series Widget. Label  |
| method), 48, 313, 317  | method), 612   |
| blowOutCellLegacy()  | blur() (AFL.automation.prepare.SampleSeriesWidget.Text   |

```
method), 633
                                                                                                                                                                 attribute), 686
blur() (AFL.automation.prepare.SampleSeriesWidget.VBobottom(AFL.automation.prepare.DeckBuilderWidget.Layout
                      method), 640
                                                                                                                                                                 attribute), 448
blur() (AFL.automation.prepare.SweepBuilderWidget.Buttbottom (AFL.automation.prepare.PrepareWidget.Layout
                      method), 654
                                                                                                                                                                 attribute), 537
blur() (AFL.automation.prepare.SweepBuilderWidget.Chebdatxom (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                                                                                                                                 attribute), 620
                      method), 661
\verb|blur()| (AFL. automation. prepare. SweepBuilder Widget. HBd \verb|bottom| (AFL. automation. prepare. SweepBuilder Widget. Layout) | (AFL. automation. prepare. Pre
                      method), 669
                                                                                                                                                                 attribute), 686
blur() (AFL.automation.prepare.SweepBuilderWidget.Labbox_style (AFL.automation.prepare.DeckBuilderWidget.HBox
                      method), 677
                                                                                                                                                                 attribute), 432
blur() (AFL.automation.prepare.SweepBuilderWidget.Textbox_style (AFL.automation.prepare.DeckBuilderWidget.VBox
                      method), 698
                                                                                                                                                                 attribute), 466
blur() (AFL.automation.prepare.SweepBuilderWidget.VBobox_style(AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                                                 attribute), 521
                      method), 705
border (AFL.automation.prepare.DeckBuilderWidget.Layolbox_style (AFL.automation.prepare.PrepareWidget.VBox
                      property), 450
                                                                                                                                                                 attribute), 560
border (AFL.automation.prepare.PrepareWidget.Layout box_style(AFL.automation.prepare.SampleSeriesWidget.HBox
                      property), 539
                                                                                                                                                                 attribute), 596
border (AFL.automation.prepare.SampleSeriesWidget.Laydbax_style (AFL.automation.prepare.SampleSeriesWidget.VBox
                      property), 622
                                                                                                                                                                 attribute), 641
border (AFL. automation. prepare. Sweep Builder Widget. Lay \textit{bordex}\_style (AFL. automation. prepare. Sweep Builder Widget. HBox) and the sum of the su
                                                                                                                                                                 attribute), 669
                      property), 688
border_bottom (AFL.automation.prepare.DeckBuilderWidgetLStonlte (AFL.automation.prepare.SweepBuilderWidget.VBox
                      attribute), 448
                                                                                                                                                                 attribute), 706
border_bottom(AFL.automation.prepare.PrepareWidget.lbnyiold_deck_object()
                      attribute), 537
                                                                                                                                                                 (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget
border_bottom(AFL.automation.prepare.SampleSeriesWidget.Layomethod), 65, 419, 471
                                                                                                                                         build_deck_object()
                      attribute), 620
border_bottom(AFL.automation.prepare.SweepBuilderWidget.Layo(AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderWidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.DeckBuilderwidget.
                      attribute), 686
                                                                                                                                                                 method), 65, 420, 471
border_left (AFL.automation.prepare.DeckBuilderWidgebflatledutdeck_object()
                      attribute), 448
                                                                                                                                                                 (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget
border_left (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                                                 method), 509
                      attribute), 537
                                                                                                                                          build_label() (AFL.automation.prepare.PrepareWidget.SampleSeriesWi
border_left (AFL.automation.prepare.SampleSeriesWidget.Layout method), 546
                      attribute), 620
                                                                                                                                         build_label() (AFL.automation.prepare.SampleSeriesWidget.SampleSer
border_left (AFL.automation.prepare.SweepBuilderWidget.Layout method), 70, 628, 646
                       attribute), 686
                                                                                                                                          bulk_cast_films() (AFL.automation.sample.CastingServer.CastingServ
border_right (AFL.automation.prepare.DeckBuilderWidget.Layout method), 89, 720, 730
                                                                                                                                         Button (class in AFL.automation.prepare.DeckBuilderWidget),
                      attribute), 448
border_right (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                         Button (class in AFL.automation.prepare.PrepareWidget),
                      attribute), 537
border_right (AFL.automation.prepare.SampleSeriesWidget.Layout492
                      attribute), 620
                                                                                                                                         Button (class in AFL.automation.prepare.SampleSeriesWidget),
border_right (AFL.automation.prepare.SweepBuilderWidget.Layou566
                      attribute), 686
                                                                                                                                         Button (class in AFL.automation.prepare.SweepBuilderWidget),
border_top(AFL.automation.prepare.DeckBuilderWidget.Layout
                                                                                                                                                               651
                      attribute), 448
                                                                                                                                          \verb|button_style| (AFL. automation. prepare. Deck Builder Widget. Button
border_top(AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                                                 attribute), 402
                                                                                                                                         \verb|button_style| (AFL. automation. prepare. Prepare Widget. Button
                      attribute), 537
border_top(AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 494
                      attribute), 620
                                                                                                                                         \verb|button_style| (AFL. automation. prepare. Sample Series Widget. Button
border_top(AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 568
```

```
button_style (AFL.automation.prepare.SweepBuilderWidgbffBnttb) (AFL.automation.EpicsADLiveProcess.AreaDetectorLive.AreaD
                                                   attribute), 653
                                                                                                                                                                                                                                                                                                                                                                 method), 26, 821
                                                                                                                                                                                                                                                                                                              ceil() (in module AFL.automation.APIServer.Driver),
C
calc_sweep() (AFL.automation.prepare.SweepBuilderWidgei.sweepBunderWidgerangometion.APIServer.DummyDriver),
                                                   method), 73, 694, 711
\verb|calc_sweep_cb()| (AFL. automation. prepare. Prepare Widgets in the dwedget L. automation. APIS erver. Dummy OT2 Driver), where the distribution of the distributio
                                                  method), 549
calc_sweep_cb() (AFL.automation.prepare.SweepBuilder Willsed SweepBuilted FW in automation.instrument.SpecScreen_Driver),
                                                  method), 73, 694, 711
\verb|calculate_bounds()| (AFL. automation. prepare. Mass Bala \textit{Rei} 1() (in module AFL. automation. sample. Casting Server), \\
                                                 method), 386
                                                                                                                                                                                                                                                                                                       cellToWaste() (AFL.automation.loading.OneSelectorBlowoutSampleCell
calibrate() (AFL.automation.loading.Sensor.Sensor
                                                                                                                                                                                                                                                                                                                                                                 method), 272
                                                   method), 52, 339
calibrate_sensor() (AFL.automation.loading.LoadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadS
                                                                                                                                                                                                                                                                                                                                                                 method), 277
                                                  method), 39, 251, 262
calibrate_sensor() (AFL.automation.loading.PneumaticPresTaWasangleCell.PnetamaticPressaliesangleCellSelectorSampleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.PressaliesangleCell.Pressal
                                                                                                                                                                                                                                                                                                                                                                method), 48, 312, 317
                                                   method), 44, 287, 291
calibrate_sensor() (AFL.automation.loading.Pneumatics lptp West.Pheuthelicsutomyteconloading.RSoXSSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCell.RSoXSolutionSampleCe
                                                                                                                                                                                                                                                                                                                                                                 method), 50, 322, 328
                                                 method), 45, 296, 301
camera_reset() (AFL.automation.instrument.NetworkCameralNeWasteComeraL.automation.loading.TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                                                                                method), 59, 370, 374
                                                   method), 30, 205
\verb|cancel()| (AFL. automation. shared. Server Discovery. Service \textbf{B} to \textbf{N} \texttt{i} \texttt{M} \texttt{ultiPosValve} \\
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                                                                                                                AFL.automation.loading.CetoniMultiPosValve),
                                                 method), 795
                                                                                                                                                                                                                                                                                                                                                                34, 226, 227
CastingServer
                                                                                                                                                                        (class
                                                                                                                                                                                                                                                                                        in
                                                                                                                                                                                                                                                                                                              change_model_callback()
                                                 AFL.automation.sample.CastingServer),
                                                                                                                                                                                                                                                                                                                                                                 (AFL. automation. shared. Data Labeler Widget. Data Labeler Widget)
                                                  89, 718, 730
                                                                                                                                                                                                                                                                                                                                                                method), 77, 741, 748
catch_sample()
                                                                                                                                   (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                              change_model_callback()
                                                   method), 383
catchToSyringe() (AFL.automation.loading.OneSelectorBlowoutSafafileCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.OntisetschaftleCutt.Ontisetschaftl
                                                                                                                                                                                                                                                                                                                                                                 method), 81, 760, 768
                                                  method), 272
catchToSyringe() (AFL.automation.loading.OneSelectorBlowsun SunfpleCell? Word State of the Color of the Color
                                                                                                                                                                                                                                                                                                                                                                (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
                                                   method), 277
catchToSyringe() (AFL.automation.loading.PushPullSelectorSampleCellPushPlatlSelectorSampleCell
                                                                                                                                                                                                                                                                                                               change_norder_callback()
                                                  method), 48, 311, 317
catchToSyringe() (AFL.automation.loading.RSoXSSolutionSample(ATLRSUNSSOLUTION) (AFL.automation.loading.RSoXSSolutionSample(ATLRSUNSSOLUTION))
                                                                                                                                                                                                                                                                                                                                                                method), 81, 760, 768
                                                   method), 50, 322, 328
catchToSyringe() (AFL.automation.loading.TwoSelectorBlowGasSantparCen.HloseleCtorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                                                                                (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
                                                  method), 59, 369, 374
catchToWaste() (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                              change_qstar_callback()
                                                  method), 272
catchToWaste() (AFL.automation.loading.OneSelectorBlowoutSampleCell!#wwstientshBted.DiffsantjienCellpeler.DiffractionLabeler
                                                                                                                                                                                                                                                                                                                                                                 method), 81, 760, 768
                                                  method), 277
catchToWaste() (AFL.automation.loading.PushPullSelect& PANDE CALL PLANDE CONTROL OF CONT
                                                                                                                                                                                                                                                                                                                                                                 method), 60, 376, 377
                                                   method), 48, 312, 317
catchToWaste() (AFL.automation.loading.RSoXSSolutionShafelectellprospectsOuthElsautomation.APIServer.APIServer.QueueDaemo
                                                                                                                                                                                                                                                                                                                                                                method), 143
                                                   method), 50, 322, 328
catchToWaste() (AFL.automation.loading.TwoSelectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwosel
                                                                                                                                                                                                                                                                                                                                                                 method), 25, 190, 192
                                                  method), 59, 370, 374
{\tt categories\_(AFL. automation. shared. Data Labeler Widget. \textbf{Checkbox} in AFL. automation. prepare. Deck Builder Widget), and the shared of the shared o
                                                   attribute), 742
categories_(AFL.automation.shared.DiffractionLabeler.Ohminkher.Caless in AFL.automation.prepare.PrepareWidget),
                                                  attribute), 763
                                                                                                                                                                                                                                                                                                              Checkbox (class in AFL.automation.prepare.SampleSeriesWidget),
```

| 574 cla  | ss_own_trait_events()  |
|--|--|
| Checkbox (class in AFL.automation.prepare.SweepBuilderWidge 659  | et), (AFL.automation.prepare.PrepareWidget.Checkbox class method), 503   |
| ChemyxConnection (class in class   | ss_own_trait_events()  |
| AFL.automation.loading.ChemyxSyringePump), 35, 228, 233  | (AFL.automation.prepare.PrepareWidget.Dropdown class method), 512        |
|  | ss_own_trait_events()  |
| AFL.automation.loading.ChemyxSyringePump), 35, 230, 232  | (AFL.automation.prepare.PrepareWidget.HBox class method), 521            |
| children (AFL.automation.prepare.DeckBuilderWidget.HBaha   |  |
| attribute), 432  | (AFL.automation.prepare.PrepareWidget.Label                              |
| children (AFL.automation.prepare.DeckBuilderWidget.VBox  | class method), 528   |
|  | ss_own_trait_events()  |
| children (AFL.automation.prepare.PrepareWidget.HBox attribute), 521  | (AFL.automation.prepare.PrepareWidget.Layout class method), 539          |
| children (AFL.automation.prepare.PrepareWidget.VBox class  | ss_own_trait_events()  |
| attribute), 560  | (AFL. automation. prepare. Prepare Widget. Text                          |
| ${\tt children}  (AFL. automation. prepare. Sample Series Widget. HBox$  |  |
|  | ss_own_trait_events()  |
| children (AFL.automation.prepare.SampleSeriesWidget.VBox attribute), 641   | (AFL.automation.prepare.PrepareWidget.VBox class method), 560            |
| ${\tt children} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. Sweep Builder Widget. {\it He} {\tt least} \ (AFL. automation. prepare. {\it He} \$ | ss_own_trait_events()  |
| attribute), 669  | (AFL. automation. prepare. Sample Series Widget. Button                  |
| children (AFL.automation.prepare.SweepBuilderWidget.VBox   |  |
|  | ss_own_trait_events()  |
| chmod() (AFL.automation.instrument.SeabreezeUVVis.Path method), 212  | (AFL.automation.prepare.SampleSeriesWidget.Checkbox class method), 577   |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Button class method), 403  | (AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 588  |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Checkboclass method), 410  | ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 596    |
| class_own_trait_events() clas  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 423  | wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 603 |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 432  | (AFL.automation.prepare.SampleSeriesWidget.Label class method), 612      |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Label class method), 439   | (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 622     |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 450  | (AFL.automation.prepare.SampleSeriesWidget.Text class method), 633       |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.Text class method), 458  | (AFL.automation.prepare.SampleSeriesWidget.VBox class method), 641       |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.DeckBuilderWidget.VBox   | (AFL.automation.prepare.SweepBuilderWidget.Button                        |
| class method), 466   | class method), 654   |
|  | ss_own_trait_events()  |
| (AFL.automation.prepare.PrepareWidget.Button class method), 495  | (AFL.automation.prepare.SweepBuilderWidget.Checkbox class method), 661   |

| <pre>class_own_trait_events()</pre>                               | class method), 596   |
|---|--|
| (AFL.automation.prepare.SweepBuilde                               | $crWidget.HR$ drass_own_traits() (AFL. automation. prepare. Sample Series Widget. Int Table 1.00 and the series Widget. Int Table 2.00 and Table 2.00 are the series Widget. Int Table 2.00 are the series Widget. In table 2.00 are the series with table 2.00 are the series Widget. In table 2.00 are the series with 2.00 are the series with table 2.00 are       |
| class method), 669  | class method), 604   |
| <pre>class_own_trait_events()</pre>                               | $\verb class_own_traits( )  (AFL. automation. prepare. Sample Series Widget. Laboration and the property of the p$ |
| (AFL. automation. prepare. Sweep Builden                          |  |
| class method), 677  | $\verb class_own_traits( )  (AFL. automation. prepare. Sample Series Widget. Layer and the sample series widget. Layer are supported by the sample series of the sample series widget. Layer are supported by the sample series will be a supported b$ |
| <pre>class_own_trait_events()</pre>                               | class method), 622   |
|   | rWidget.La <b>çbas</b> s_own_traits()(AFL.automation.prepare.SampleSeriesWidget.Text   |
| class method), 688  | class method), 633   |
| <pre>class_own_trait_events()</pre>                               | class_own_traits()(AFL.automation.prepare.SampleSeriesWidget.VBa   |
| (AFL.automation.prepare.SweepBuilde                               |  |
| class method), 698  | class_own_traits()(AFL.automation.prepare.SweepBuilderWidget.But   |
| <pre>class_own_trait_events()</pre>                               | class method), 654   |
|   | erWidget.VBthass_own_traits() (AFL.automation.prepare.SweepBuilderWidget.Che   |
| class method), 706  | class method), 661   |
|   | e.DeckBuilderWidgewButtoaits() (AFL.automation.prepare.SweepBuilderWidget.HBd  |
| class method), 403  | class method), 670   |
|   | e.DeckBuilder <b>\Kis</b> lg <b>ew6]herkho</b> s() (AFL.automation.prepare.SweepBuilderWidget.Lab  |
| class method), 410  | class method), 677   |
|   | e.DeckBuilderWislg <b>ewD_ropdows</b> () (AFL.automation.prepare.SweepBuilderWidget.Lay  |
| class method), 423  | class method), 688   |
|   | e.DeckBuilderWisgewhlBaxaits() (AFL.automation.prepare.SweepBuilderWidget.Text   |
| class method), 432  | class method), 698   |
|   | e.DeckBuilderWislgowhabalaits() (AFL.automation.prepare.SweepBuilderWidget.VBo   |
| class method), 439  | class method), 706   |
| class_own_traits() (AFL.automation.preparation class method), 450 | e.DeckBuu <b>лavassiger:nav</b> go <b>m</b> ames()<br>(AFL.automation.prepare.DeckBuilderWidget.Button   |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 458  | class_trait_names()  |
|   | e.DeckBuilderWidget(ABdxautomation.prepare.DeckBuilderWidget.Checkbox  |
| class method), 466  | class method), 411   |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 495  | (AFL.automation.prepare.DeckBuilderWidget.Dropdown   |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 503  | class_trait_names()  |
| **  | e.PrepareWidget.DropAbWautomation.prepare.DeckBuilderWidget.HBox   |
| class method), 512  | class method), 432   |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 521  | (AFL.automation.prepare.DeckBuilderWidget.Label  |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 528  | class_trait_names()  |
|   | e.PrepareWidget.Lay <b>oA</b> FL.automation.prepare.DeckBuilderWidget.Layout   |
| class method), 539  | class method), 450   |
| class_own_traits() (AFL.automation.prepara                        |  |
| class method), 552  | (AFL.automation.prepare.DeckBuilderWidget.Text   |
| class_own_traits()(AFL.automation.prepara                         |  |
| class method), 560  | class_trait_names()  |
|   | e.SampleSeriesWidget,ABultanutomation.prepare.DeckBuilderWidget.VBox   |
| class method), 569  | class method), 466   |
| class_own_traits() (AFL.automation.prepara                        |  |
| class method), 577  | (AFL.automation.prepare.PrepareWidget.Button   |
| class_own_traits() (AFL.automation.prepare                        |  |
| class method), 588  | class_trait_names()  |
|   | e.SampleSeriesWidgetAHIonutomation.prepare.PrepareWidget.Checkbox  |
|   |  |

| class method), 503  | class method), 670   |
|---|--|
| <pre>class_trait_names()</pre>                                    | <pre>class_trait_names()</pre>   |
| (AFL.automation.prepare.PrepareWidget.Dropdo                      | own (AFL.automation.prepare.SweepBuilderWidget.Label                             |
| class method), 512  | class method), 678   |
| <pre>class_trait_names()</pre>                                    | <pre>class_trait_names()</pre>   |
| (AFL.automation.prepare.PrepareWidget.HBox                        | (AFL.automation.prepare.SweepBuilderWidget.Layout                                |
| class method), 521  | class method), 688   |
| class_trait_names()   | <pre>class_trait_names()</pre>   |
| (AFL.automation.prepare.PrepareWidget.Label                       | (AFL.automation.prepare.SweepBuilderWidget.Text                                  |
| class method), 529  | class method), 698   |
| class_trait_names()   | class_trait_names()  |
| (AFL.automation.prepare.PrepareWidget.Layout                      |  |
| class method), 539  | class method), 706   |
| class_trait_names()   | class_traits()(AFL.automation.prepare.DeckBuilderWidget.Button                   |
| (AFL.automation.prepare.PrepareWidget.Text                        | class method), 403   |
| class method), 552  | class_traits()(AFL.automation.prepare.DeckBuilderWidget.Checkbox                 |
| class_trait_names()   | class method), 411   |
| (AFL.automation.prepare.PrepareWidget.VBox                        |  |
| class method), 560  | class method), 424   |
| class_trait_names()   | class_traits()(AFL.automation.prepare.DeckBuilderWidget.HBox                     |
| (AFL.automation.prepare.SampleSeriesWidget.B                      |  |
| class method), 569  | class_traits()(AFL.automation.prepare.DeckBuilderWidget.Label                    |
| class_trait_names()   | class method), 440   |
|   | heddss_traits()(AFL.automation.prepare.DeckBuilderWidget.Layout                  |
| class method), 577  | class method), 450   |
| class_trait_names()   | class_traits() (AFL.automation.prepare.DeckBuilderWidget.Text                    |
| (AFL.automation.prepare.SampleSeriesWidget.F.                     |  |
| class method), 588  | class_traits()(AFL.automation.prepare.DeckBuilderWidget.VBox                     |
| class_trait_names()   | class method), 466   |
|   | Bakass_traits() (AFL.automation.prepare.PrepareWidget.Button                     |
| class method), 596  | class method), 495   |
| class_trait_names()   | class_traits() (AFL.automation.prepare.PrepareWidget.Checkbox                    |
| (AFL.automation.prepare.SampleSeriesWidget.In                     |  |
| class method), 604  |  |
| class_trait_names()   | class_traits() (AFL.automation.prepare.PrepareWidget.Dropdown class method), 513 |
| **  |  |
|   | abdlass_traits() (AFL.automation.prepare.PrepareWidget.HBox class method), 521   |
| class method), 612  |  |
| class_trait_names() (AFL.automation.prepare.SampleSeriesWidget.La | class_traits() (AFL.automation.prepare.PrepareWidget.Label                       |
| 1 1 1   |  |
| class method), 622  | class_traits() (AFL.automation.prepare.PrepareWidget.Layout                      |
| class_trait_names()   | class method), 539   |
|   | extlass_traits()(AFL.automation.prepare.PrepareWidget.Text                       |
| class method), 633  | class method), 552   |
| class_trait_names()   | class_traits() (AFL.automation.prepare.PrepareWidget.VBox                        |
| (AFL.automation.prepare.SampleSeriesWidget.V.                     |  |
| class method), 641  | class_traits() (AFL.automation.prepare.SampleSeriesWidget.Button                 |
| class_trait_names()   | class method), 569   |
|   | Cutlouss_traits()(AFL.automation.prepare.SampleSeriesWidget.Checkbo              |
| class method), 654  | class method), 577   |
| class_trait_names()   | class_traits()(AFL.automation.prepare.SampleSeriesWidget.FloatTe.                |
| (AFL.automation.prepare.SweepBuilderWidget.C                      |  |
| class method), 662  | class_traits()(AFL.automation.prepare.SampleSeriesWidget.HBox                    |
| class_trait_names()   | class method), 596   |
| (AFL.automation.prepare.SweepBuilderWidget.H                      | <pre>####################################</pre>                                  |

- class method), 604 method), 482
- class\_traits() (AFL.automation.prepare.SampleSeriesWillgertLatiestory() (AFL.automation.prepare.OT2Client.OT2Client class method), 612 method), 485
- class\_traits() (AFL.automation.prepare.SampleSeriesWidget.Client class method), 622 method), 585
- class\_traits() (AFL.automation.prepare.SampleSeriesWidlgerTehistory() (AFL.automation.sample.CastingServer.Client class method), 633 method), 724
- class\_traits() (AFL.automation.prepare.SampleSeriesWidlgeartVRistory() (AFL.automation.sample.CastingServer.OT2Client class method), 641 method), 729
- class\_traits()(AFL.automation.prepare.SweepBuilderWillgatrBottoput() (in module class method), 654

  AFL.automation.shared.widgetui), 814
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgatrChaekbe(x) (AFL.automation.APIServer.
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgarHqueue() (AFL.automation.APIServer.Client.Client class method), 670 method), 20, 161, 164
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgarLqbelue() (AFL.automation.loading.LoadStopperDriver.Client class method), 678 method), 246
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgarLayente() (AFL.automation.prepare.DeckBuilderWidget.Client class method), 418
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgatrTexteue() (AFL.automation.prepare.OT2Client.Client class method), 698 method), 482
- class\_traits() (AFL.automation.prepare.SweepBuilderWillgarV@oeue() (AFL.automation.prepare.OT2Client.OT2Client class method), 706 method), 485
- clear() (AFL.automation.APIServer.Driver.PersistentConfglear\_queue() (AFL.automation.prepare.SampleSeriesWidget.Client method), 171 method), 585
- clear() (AFL.automation.APIServer.QueueDaemon.DataTeldarmqueue() (AFL.automation.sample.CastingServer.Client method), 189 method), 724
- clear() (AFL.automation.loading.OneSelectorBlowoutSanqle&rllqluquel(dicAFL.automation.sample.CastingServer.OT2Client method), 279 method), 729
- clear() (AFL.automation.loading.PneumaticPressureSamp**dleC**ell(**Aefdudadioc**mation.APIServer.APIServer.Flask atmethod), 289 tribute), 130
- clear() (AFL.automation.loading.PneumaticSampleCell.defhiutdd() (AFL.automation.prepare.DeckBuilderWidget.Button method), 299 method), 402
- clear() (AFL.automation.loading.PushPullSelectorSampleCleik&ffu(MHLcautomation.prepare.PrepareWidget.Button method), 315 method), 495
- clear() (AFL.automation.loading.RSoXSSolutionSampleCellixeKQ)(AFL.automation.prepare.SampleSeriesWidget.Button method), 326 method), 569
- clear() (AFL.automation.loading.TwoSelectorBlowoutSamplaCkl()defAtilLationation.prepare.SweepBuilderWidget.Button method), 372 method), 653
- clear() (AFL.automation.shared.DatasetWidget.defaultdic€lient (class in AFL.automation.APIServer.Client), 19, method), 755 159, 164
- clear() (AFL.automation.shared.PersistentConfig.MutableMappritgclass in AFL.automation.EpicsADLiveProcess.Client), method), 772 26, 821, 822
- clear() (AFL.automation.shared.PersistentConfig.PersistentConfig.Class in AFL.automation.loading.LoadStopperDriver), method), 775 244
- clear\_history() (AFL.automation.APIServer.APIS
- clear\_history() (AFL.automation.APIServer.Client.Clie&lient (class in AFL.automation.prepare.OT2Client), method), 20, 161, 165 480
- clear\_history() (AFL.automation.loading.LoadStopperDivientEllehass in AFL.automation.prepare.SampleSeriesWidget), method), 246 582
- clear\_history() (AFL.automation.prepare.DeckBuilderWildgenCliebutss in AFL.automation.sample.CastingServer), method), 418 721
- clear\_history()(AFL.automation.prepare.OT2Client.Client.construct\_ui() (in module

- AFL.automation.shared.widgetui), 88, 814, close() (AFL.automation.prepare.SampleSeriesWidget.Text method), 633
- close() (AFL.automation.APIServer.APIServer.FileHandlaclose() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 104 method), 641
- close() (AFL.automation.APIServer.APIServer.SMTPHandlerse() (AFL.automation.prepare.SweepBuilderWidget.Button method), 146 method), 654
- close() (AFL.automation.APIServer.APIServer.Zeroconf close() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 157 method), 662
- close() (AFL.automation.prepare.DeckBuilderWidget.Buttohose() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 403 method), 670
- close() (AFL.automation.prepare.DeckBuilderWidget.Che**cklusse**() (AFL.automation.prepare.SweepBuilderWidget.Label method), 411 method), 678
- close() (AFL.automation.prepare.DeckBuilderWidget.Dropdlose() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 424 method), 688
- close() (AFL.automation.prepare.DeckBuilderWidget.HBalose() (AFL.automation.prepare.SweepBuilderWidget.Text method), 432 method), 698
- close() (AFL.automation.prepare.DeckBuilderWidget.Labellose() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 440 method), 706
- close() (AFL.automation.prepare.DeckBuilderWidget.Lay@ltose() (AFL.automation.shared.ServerDiscovery.Zeroconf method), 450 method), 808
- close() (AFL.automation.prepare.DeckBuilderWidget.Textclose\_all() (AFL.automation.prepare.DeckBuilderWidget.Button method), 458 class method), 403
- close() (AFL.automation.prepare.DeckBuilderWidget.VBoxlose\_all() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 466 class method), 411
- close() (AFL.automation.prepare.PrepareWidget.Button close\_all() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 495 class method), 424
- close() (AFL.automation.prepare.PrepareWidget.Checkboxclose\_all() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 503 class method), 432
- close() (AFL.automation.prepare.PrepareWidget.Dropdowolose\_all() (AFL.automation.prepare.DeckBuilderWidget.Label method), 513 class method), 440
- close() (AFL.automation.prepare.PrepareWidget.HBox close\_all() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 521 class method), 451
- close() (AFL.automation.prepare.PrepareWidget.Label close\_all() (AFL.automation.prepare.DeckBuilderWidget.Text method), 529 class method), 458
- close() (AFL.automation.prepare.PrepareWidget.Layout close\_all() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 539 class method), 466
- close() (AFL.automation.prepare.PrepareWidget.Text close\_all() (AFL.automation.prepare.PrepareWidget.Button method), 552 class method), 496
- close() (AFL.automation.prepare.PrepareWidget.VBox close\_all() (AFL.automation.prepare.PrepareWidget.Checkbox method), 560 class method), 503
- close() (AFL.automation.prepare.SampleSeriesWidget.Butchose\_all() (AFL.automation.prepare.PrepareWidget.Dropdown method), 569 class method), 513
- close() (AFL.automation.prepare.SampleSeriesWidget.Checklosse\_all() (AFL.automation.prepare.PrepareWidget.HBox method), 577 class method), 521
- close() (AFL.automation.prepare.SampleSeriesWidget.FloalEse\_all() (AFL.automation.prepare.PrepareWidget.Label method), 588 class method), 529
- close() (AFL.automation.prepare.SampleSeriesWidget.HBolose\_all() (AFL.automation.prepare.PrepareWidget.Layout method), 596 class method), 540
- close() (AFL.automation.prepare.SampleSeriesWidget.IntEdose\_all() (AFL.automation.prepare.PrepareWidget.Text method), 604 class method), 552
- close() (AFL.automation.prepare.SampleSeriesWidget.Labelose\_all() (AFL.automation.prepare.PrepareWidget.VBox method), 612 class method), 560
- close() (AFL.automation.prepare.SampleSeriesWidget.Layakase\_all() (AFL.automation.prepare.SampleSeriesWidget.Button method), 622 class method), 569

- close\_all() (AFL.automation.prepare.SampleSeriesWidgetchte(AFL.xautomation.prepare.DeckBuilderWidget.HBox class method), 577 attribute), 432
- close\_all() (AFL.automation.prepare.SampleSeriesWidgeroRiko(ATEktautomation.prepare.DeckBuilderWidget.Label class method), 588 attribute), 440
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetoHR (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 596 attribute), 451
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetchntTe(AFL.automation.prepare.DeckBuilderWidget.Text class method), 604 attribute), 458
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetcharb(AFL.automation.prepare.DeckBuilderWidget.VBox class method), 612 attribute), 466
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetchmy(AuFL.automation.prepare.PrepareWidget.Button atclass method), 623 tribute), 496
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetoFarx(AFL.automation.prepare.PrepareWidget.Checkbox class method), 633 attribute), 503
- close\_all() (AFL.automation.prepare.SampleSeriesWidgetoMBAFL.automation.prepare.PrepareWidget.Dropdown class method), 641 attribute), 513
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetblatt(AFL.automation.prepare.PrepareWidget.HBox atclass method), 654 tribute), 521
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetMihe(AFbxautomation.prepare.PrepareWidget.Label atclass method), 662 tribute), 529
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetMtBox (AFL.automation.prepare.PrepareWidget.Layout class method), 670 attribute), 540
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetMmbelAFL.automation.prepare.PrepareWidget.Text atclass method), 678 tribute), 552
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetoImny&AFL.automation.prepare.PrepareWidget.VBox atclass method), 689 tribute), 560
- close\_all() (AFL.automation.prepare.SweepBuilderWidget) (AFL.automation.prepare.SampleSeriesWidget.Button class method), 698 attribute), 569
- close\_all() (AFL.automation.prepare.SweepBuilderWidgetMiR(AFL.automation.prepare.SampleSeriesWidget.Checkbox class method), 706 attribute), 577
- closeConnection() (AFL.automation.loading.ChemyxSyriroganP(AhrJLGhuennyaCionspretiane.SampleSeriesWidget.FloatText method), 36, 229, 233 attribute), 588
- collect() (AFL.automation.instrument.FileCamera.FileCam
- collect() (AFL.automation.instrument.NetworkCamera.NetworkCEffnentomation.prepare.SampleSeriesWidget.IntText method), 30, 205 attribute), 604
- collect() (AFL.automation.instrument.SeabreezeUVVis.SeabreeZeUVis.SeabreeZeUVVis.SeabreeZeUVVis.SeabreeZeUVVis.
- collectContinuous() comm (AFL.automation.prepare.SampleSeriesWidget.Layout (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVistribute), 623 method), 30, 217, 218 comm (AFL.automation.prepare.SampleSeriesWidget.Text
- collectSingleSpectrum() attribute), 633
  (AFL.automation.instrument.SeabreezeUVVis.Seabonne(NVV)isutomation.prepare.SampleSeriesWidget.VBox method), 30, 217, 218 attribute), 641
- combine\_vars() (AFL.automation.shared.DatasetWidget.DottmctWillgettomation.prepare.SweepBuilderWidget.Button method), 79, 751, 757 attribute), 654
- combine\_vars() (AFL.automation.shared.DatasetWidget.**Rotun**(AWillgattoMudidn.prepare.SweepBuilderWidget.Checkbox method), 80, 753, 758 attribute), 662
- comm (AFL.automation.prepare.DeckBuilderWidget.Button comm (AFL.automation.prepare.SweepBuilderWidget.HBox attribute), 403 attribute), 670
- comm (AFL.automation.prepare.DeckBuilderWidget.Checkb@omm (AFL.automation.prepare.SweepBuilderWidget.Label attribute), 411 attribute), 678
- comm (AFL.automation.prepare.DeckBuilderWidget.Dropdowomm (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 424 attribute), 689

| comm (AFL.automation.prepare.SweepBuilderWidget.Text attribute), 698   | copy() (AFL.automation.loading.PneumaticSampleCell.defaultdict method), 299              |  |  |
|--|--|--|--|
| comm (AFL.automation.prepare.SweepBuilderWidget.VBox attribute), 706   | x copy() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdi<br>method), 315    |  |  |
|  | copy() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict method), 326          |  |  |
| ComponentDB (class in AFL.automation.prepare), 382   | copy() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultd                     |  |  |
| composition_click_callback()   | method), 372   |  |  |
| (AFL.automation.shared.DataLabelerWidget.Dat<br>method), 77, 740, 748  | talapse (er Mildkeautomation.prepare.Component.Component<br>method), 64, 396, 398        |  |  |
| <pre>composition_click_callback()</pre>  | copy() (AFL.automation.prepare.factory.Solution  |  |  |
| (AFL.automation.shared.DatasetWidget.Dataset     |  |  |  |
| method), 79, 751, 757  | copy() (AFL.automation.prepare.MassBalance method),                                      |  |  |
| <pre>compositionSweepFactory() (in module</pre>  | 385  |  |  |
| AFL.automation.prepare), 381   | copy() (AFL.automation.prepare.Solute method), 389                                       |  |  |
|  | copy() (AFL.automation.prepare.Solution method), 391                                     |  |  |
|  | copy() (AFL.automation.prepare.Solvent method), 393                                      |  |  |
| 716  | copy() (AFL.automation.shared.DatasetWidget.defaultdict                                  |  |  |
| <pre>compute_checksum() (AFL.automation.loading.UltimusV</pre>   |  |  |  |
| method), 60, 376, 377  | CORS (class in AFL.automation.APIServer.APIServer),                                      |  |  |
| concentration (AFL.automation.prepare.factory.Solution   |  |  |  |
| property), 715   | create_access_token() (in module   |  |  |
| concentration (AFL.automation.prepare.Solution   | AFL.automation.APIServer.APIServer), 91  |  |  |
| property), 392   | <pre>create_expanded_button()</pre>  |  |  |
| config (AFL.automation.APIServer.APIServer.Flask at-   | (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidge                               |  |  |
| tribute), 113  | method), 65, 421, 471  |  |  |
| config_class (AFL.automation.APIServer.APIServer.Fla   |  |  |  |
| attribute), 111  | (AFL.automation.APIServer.APIServer.Flask  |  |  |
| constrain_samples_conc()   | method), 116   |  |  |
| (AFL.automation.prepare.MassBalance  | create_jinja_environment()   |  |  |
| method), 386   | (AFL.automation.APIServer.APIServer.Flask  |  |  |
| contains() (AFL.automation.prepare.factory.Solution  | method), 116   |  |  |
| method), 715   | create_queue() (AFL.automation.APIServer.APIServer                                       |  |  |
| contains() (AFL.automation.prepare.Solution  | method), 18, 99, 157   |  |  |
| method), 391   | create_refresh_token() (in module  |  |  |
| context_processor()  | AFL.automation.APIServer.APIServer), 92  |  |  |
| (AFL.automation.APIServer.APIServer.Flask  | create_url_adapter()   |  |  |
| method), 127   | (AFL.automation.APIServer.APIServer.Flask  |  |  |
|  |  |  |  |
| continuous_update (AFL.automation.prepare.DeckBuila attribute), 457  |  |  |  |
|  | createLock() (AFL.automation.APIServer.APIServer.FileHandler                             |  |  |
| continuous_update (AFL.automation.prepare.PrepareW   |  |  |  |
| attribute), 551  | createLock() (AFL.automation.APIServer.APIServer.SMTPHandler                             |  |  |
| continuous_update(AFL.automation.prepare.SampleSet   |  |  |  |
| attribute), 587  | cross_validation_lock  |  |  |
|  | riesWidget.(AffExautomation.prepare.DeckBuilderWidget.Button                             |  |  |
| attribute), 603  | property), 403   |  |  |
| continuous_update(AFL.automation.prepare.SampleSet   | v .  |  |  |
| attribute), 632  | (AFL.automation.prepare.DeckBuilderWidget.Checkbox                                       |  |  |
| continuous_update(AFL.automation.prepare.SweepBuit   | * * * * * ·  |  |  |
| attribute), 697  | cross_validation_lock  |  |  |
| method), 279   | pleCell.def <b>AMilia</b> utomation.prepare.DeckBuilderWidget.Dropdown<br>property), 424 |  |  |
| $\verb"copy"()" (AFL. automation. loading. Pneumatic Pressure Sample and the property of the prop$ | oleCollsdefaillidiation_lock   |  |  |
| method), 289   | (AFL.automation.prepare.DeckBuilderWidget.HBox   |  |  |

| property), 432   |                 | property), 612   |
|--|-----------------|--|
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL. automation. prepare. Deck Builder Widget. Landing and the property of  | abel            | (AFL.automation.prepare.SampleSeriesWidget.Layout  |
| property), 440   |                 | property), 623   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL. automation. prepare. Deck Builder Widget. Landing and the property of  | ayout           | (AFL.automation.prepare.SampleSeriesWidget.Text  |
| property), 451   |                 | property), 633   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL. automation. prepare. Deck Builder Widget. Te   | ext             | (AFL.automation.prepare.SampleSeriesWidget.VBox  |
| property), 458   |                 | property), 641   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL. automation. prepare. Deck Builder Widget. VI   | Box             | (AFL. automation. prepare. Sweep Builder Widget. Button  |
| property), 466   |                 | property), 654   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL. automation. prepare. Prepare Widget. Button  | !               | (AFL. automation. prepare. Sweep Builder Widget. Checkbox  |
| property), 496   |                 | property), 662   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL.automation.prepare.PrepareWidget.Checkl   | box             | (AFL.automation.prepare.SweepBuilderWidget.HBox  |
| property), 503   |                 | property), 670   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL.automation.prepare.PrepareWidget.Dropde   | own             | (AFL.automation.prepare.SweepBuilderWidget.Label   |
| property), 513   |                 | property), 678   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL.automation.prepare.PrepareWidget.HBox   |                 | (AFL.automation.prepare.SweepBuilderWidget.Layout  |
| property), 521   |                 | property), 689   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL.automation.prepare.PrepareWidget.Label  |                 | (AFL.automation.prepare.SweepBuilderWidget.Text  |
| property), 529   |                 | property), 698   |
| cross_validation_lock  | cross_          | validation_lock  |
| (AFL.automation.prepare.PrepareWidget.Layout   | t               | (AFL.automation.prepare.SweepBuilderWidget.VBox  |
| property), 540   |                 | property), 706   |
| cross_validation_lock  | cwd() (         | AFL.automation.instrument.SeabreezeUVVis.Path  |
| (AFL.automation.prepare.PrepareWidget.Text   |                 | class method), 211   |
| property), 552   | _               |  |
| cross_validation_lock  | D               |  |
| (AFL. automation. prepare. Prepare Widget. VBox  | daemon          | (AFL.automation.APIServer.APIServer.QueueDaemon  |
| property), 560   |                 | property), 143   |
| cross_validation_lock  | daemon          | (AFL.automation.APIServer.QueueDaemon.QueueDaemon  |
| (AFL. automation. prepare. Sample Series Widget. B   |                 | property), 190   |
| property), 569   | daemon          | (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDae  |
| cross_validation_lock  |                 | property), 825   |
| (AFL. automation. prepare. Sample Series Widget. Color for the prepared of t | Charlet Moon    | (AFL. automation. loading. Load Stopper Driver. Sensor Polling Thread Stopper Driver. Sensor Polling Threa |
| property), 577   |                 | property), 254   |
| cross_validation_lock  | daemon          | (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1   |
| (AFL. automation. prepare. Sample Series Widget. F   | FloatText       | property), 257   |
| property), 588   | daemon          | (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2   |
| cross_validation_lock  |                 | property), 260   |
| (AFL. automation. prepare. Sample Series Widget. However, and the substitution of th | <i>Ba</i> remon | (AFL.automation.loading.Sensor.DummySensor1  |
| property), 596   |                 | property), 334   |
| cross_validation_lock  | daemon          | (AFL.automation.loading.Sensor.DummySensor2  |
| (AFL. automation. prepare. Sample Series Widget. In the prepare of the prepare  |                 | property), 338   |
| property), 604   | daemon          | (AFL.automation.loading.SensorCallbackThread.SensorCallbackT   |
| cross_validation_lock  |                 | property), 343   |
| (AFL. automation. prepare. Sample Series Widget. L   | abel            | •  |

| ${\tt daemon} (AFL. automation. loading. Sensor Callback Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt Callback} Thread. Since {\tt daemon} (AFL. automation. loading. Sensor {\tt daemon} (AFL. automation. loading. load$  | mpleThre              | shetH&B), 585                              |                          |                      |
|---|-----------------------|--|--------------------------|----------------------|
|   | _                     | (AFL.automation.sample                     | .CastingServer.Clien     | nt                   |
| ${\tt daemon} (AFL. automation. loading. Sensor Callback Thread. States and the sensor Callback Thread. States are sensor for the sensor$   |                       |  |                          |                      |
|   |                       | (AFL.automation.sample.                    | CastingServer.OT2C       | lient                |
| ${\tt daemon} (AFL. automation. loading. Sensor Callback Thread. States and the sensor Callback Thread. States are sensor for the sensor$   |                       |  |                          |                      |
|   |                       | ıss in AFL.automation.prep                 | pare), 382               |                      |
| ${\tt daemon} (AFL. automation. loading. Sensor Polling Thread. Sen {\tt Sensor} Polling Thread$ |                       | _  | `                        | in                   |
| property), 357  |                       | AFL.automation.prepare.                    | DeckBuilderWidget)       | ,                    |
| ${\tt daemon}(AFL. automation. shared. Server Discovery. Run Threat Control of the Control of th$   |                       | 65, 419, 471                               |                          |                      |
|   |                       | _  | \                        | in                   |
| daemon (AFL.automation.shared.ServerDiscovery.ServiceBrandsproperty), 795   | rowser                | AFL.automation.prepare.                    | PrepareWidget),          |                      |
| data (AFL.automation.loading.PneumaticPressureSampleCo  | a H c RiRavivi        |  | (class                   | in                   |
| property), 43, 286, 290   |                       | AFL.automation.prepare.                    | •                        |                      |
| DataLabelerModel (class in  |                       | 65, 420, 471                               | seekBuilder Widger)      | ,                    |
| · · · · · · · · · · · · · · · · · · ·   | DeckBui               | lderWidget_View                            | (class                   | in                   |
| 77, 738, 748  |                       | AFL.automation.prepare.l                   | *                        |                      |
| DataLabelerView (class in   |                       | 65, 420, 471                               | seekBuilder Widger)      | ,                    |
|   | decode(               | ) (AFL.automation.instrun                  | nent SeabreezeUVVi       | s Ea                 |
| 77, 738, 748  | accouc <sub>(,</sub>  | class method), 208                         | ieni.Sedoreegee v vii    | Lq                   |
| DataLabelerWidget (class in c   | decode_l              | key_loader()                               |                          |                      |
| AFL.automation.shared.DataLabelerWidget), 76, 739, 747  |                       | (AFL.automation.APIServ method), 138       | ver.APIServer.JWTM       | lanager              |
| dataset (AFL.automation.shared.DatasetWidget  |                       | / ·  | nation APISarvar AP      | ISarvar SarvicaInfo  |
| property), 79, 752, 757   | msen <del>u</del> nan | attribute), 150                            | anon.Al Iserver.Al       | iserver.serviceinjo  |
| DatasetWidget (class in c   | decoded <u>.</u>      | _properties(AFL.autom                      | ation.shared.Server      | Discovery.AsyncSei   |
| AFL. automation. shared. Dataset Widget),   |                       | attribute), 782                            |                          |                      |
| 78, 749, 756  |                       | _properties(AFL.autom                      | ation.shared.Server      | Discovery.ServiceIn  |
| DatasetWidget_Model (class in   |                       | attribute), 800                            |                          |                      |
| AFL.automation.shared.DatasetWidget), 79, 752, 757  | default <sub>.</sub>  | _config(AFL.automation attribute), 112     | APIServer.APIServ        | er.Flask             |
|   | default               | _factory(AFL.automatio                     | on loading OneSelec      | torRlowoutSample(    |
| AFL.automation.shared.DatasetWidget),   | acraure.              | attribute), 279                            | n.iouuing.Oneseice       | iorBiowouisampieC    |
| 80, 753, 758  | default.              | _factory(AFL.automatio                     | n.loading.Pneumati       | cPressureSampleCe    |
| DataTrashcan (class in  |                       | attribute), 289                            |                          |                      |
| AFL.automation.APIServer.QueueDaemon), c  | default <sub>.</sub>  | _factory (AFL.automatio<br>attribute), 299 | n.loading.Pneumati       | cSampleCell.defaul   |
| deactivate_temp() (AFL.automation.prepare.Dummy_Ol  | ATA FAMINTA           | * *  | on loading PushPull      | SelectorSampleCell   |
| method), 66, 477, 479   |                       | attribute), 315                            | n.iouuing.i usiii uii.   | ociccioi sumpie ceii |
| debug (AFL.automation.APIServer.APIServer.Flask o   |                       | · · · · · · · · · · · · · · · · · · ·      | on loading RSaXSSa       | lutionSampleCell de  |
| property), 117  | ucruurt.              | attribute), 326                            | 11.10441115.1150215501   | initorisampie een.ac |
| debug() (AFL.automation.APIServer.APIServer.APIServer   | default               |  | on loading TwoSelec      | torRlowoutSampleC    |
| method), 19, 100, 159   | ucruurt.              | attribute), 372                            | n.ioaaing.1woscicei      | iorBiowouisampiee    |
| debug() (AFL.automation.APIServer.Client.Client   | default               | * * * * * * * * * * * * * * * * * * *      | on shared DatasetWi      | doet defaultdict     |
| method), 20, 161, 165   | acraure.              | attribute), 755                            | Tilstica ca. Baraser + v | agenaejamnarer       |
| debug() (AFL.automation.loading.LoadStopperDriver.Clien   | nle fault             |  | 22                       | in                   |
| method), 246  |                       | AFL.automation.loading.                    |                          |                      |
| debug() (AFL.automation.prepare.DeckBuilderWidget.Clien   |                       | 279  | Juege ie eio i Biowoui   | sample cell);        |
|   | default               |  | 22                       | in                   |
| debug() (AFL.automation.prepare.OT2Client.Client  |                       | AFL.automation.loading.l                   |                          |                      |
| method), 482  |                       | 288  |                          | pic ceiij,           |
| debug() (AFL.automation.prepare.OT2Client.OT2Client   | defaul+               |  | SS                       | in                   |
| method), 485  |                       | AFL.automation.loading.l                   |                          |                      |
| debug() (AFL.automation.prepare.SampleSeriesWidget.Clie   |                       | 298  | proce                    | //                   |

```
defaultdict
                                                                                        (class
                                                                                                                                                                                               method), 22, 168, 172
                                                                                                                                                        in
                          AFL.automation.loading.PushPullSelectorSampleCoppsit_obj() (AFL.automation.APIServer.DummyDriver.Driver
                                                                                                                                                                                               method), 175
defaultdict
                                                                                                                                                        in deposit_obj() (AFL.automation.APIServer.DummyDriver.DummyDriver
                                                                                        (class
                          AFL.automation.loading.RSoXSSolutionSampleCell),
                                                                                                                                                                                               method), 178
                                                                                                                                                                    deposit_obj() (AFL.automation.APIServer.DummyOT2Driver.Driver
defaultdict
                                                                                        (class
                                                                                                                                                        in
                                                                                                                                                                                               method), 182
                          AFL.automation.loading.TwoSelectorBlowoutSample(Osli)t_obj() (AFL.automation.APIServer.DummyOT2Driver.DummyDelectorBlowoutSample(Osli)t_obj() (AFL.automation.APIServer.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driver.DummyOT2Driv
                           371
                                                                                                                                                                                               method), 185
defaultdict
                                                                                                                                                        in deposit_obj()(AFL.automation.instrument.DummySAS.Driver
                                                                                        (class
                          AFL.automation.shared.DatasetWidget),
                                                                                                                                                                                               method), 195
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.DummySAS.DummySAS
defaults (AFL.automation.APIServer.DummyDriver.DummyDriver method), 197
                          attribute), 23, 178, 179
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.I22SAXS.Driver
defaults (AFL.automation.APIServer.DummyOT2Driver.DummyDrimenthod), 201
                           attribute), 24, 184, 186
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.I22SAXS.I22SAXS
defaults (AFL.automation.instrument.DummySAS.DummySAS
                                                                                                                                                                                               method), 203
                           attribute), 28, 197, 198
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.SeabreezeUVVis.Driver
defaults (AFL.automation.instrument.I22SAXS.I22SAXS
                                                                                                                                                                                               method), 207
                           attribute), 29, 203, 204
                                                                                                                                                                   deposit_obj() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeU
defaults (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVWithod), 217
                          attribute), 30, 216, 218
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.SpecScreen_Driver.Driver
defaults (AFL.automation.loading.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriv
                           attribute), 39, 251, 261
                                                                                                                                                                    deposit_obj() (AFL.automation.instrument.SpecScreen_Driver.SpecScre
defaults (AFL.automation.loading.OneSelectorBlowoutSampleCell. OnathSedle)ctDxBlowoutSampleCell
                           attribute), 272
                                                                                                                                                                   deposit_obj() (AFL.automation.loading.LoadStopperDriver.Client
defaults (AFL.automation.loading.OneSelectorBlowoutSampleCell. TwedSedle; (D4BlowoutSampleCell
                                                                                                                                                                    {\tt deposit\_obj()} \ (AFL. automation. loading. Load Stopper Driver. Driver
                           attribute), 276
defaults (AFL.automation.loading.PneumaticPressureSampleCell.PmeumaticPressureSampleCell
                           attribute), 43, 286, 290
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.LoadStopperDriver.LoadStoppe
defaults (AFL.automation.loading.PneumaticSampleCell.PneumatioSathplie)Cell1
                           attribute), 44, 296, 300
                                                                                                                                                                    deposit\_obj() (AFL. automation. loading. One Selector Blowout Sample Celling and Selector Blowout Selector Blowout Sample Celling and Selector Blowout Selector Blowo
defaults (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXfixahud)n.SampleCell
                                                                                                                                                                   deposit_obj() (AFL.automation.loading.OneSelectorBlowoutSampleCell
                           attribute), 49, 322, 327
defaults (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TweoSelectOrBlowoutSampleCell
                          attribute), 59, 369, 373
                                                                                                                                                                   deposit\_obj() (AFL.automation.loading.OneSelectorBlowoutSampleCell
defaults (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OTRethord)e277
                           attribute), 66, 476, 478
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.PneumaticPressureSampleCell...
{\tt defaults} \ (AFL. automation. sample. Casting Server. Casting Server
                                                                                                                                                                                               method), 283
                           attribute), 89, 720, 730
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.PneumaticPressureSampleCell...
defaults (AFL.automation.sample_env.TemperatureDeck.TemperatumeDeckl), 287
                           attribute), 75, 735, 736
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.PneumaticSampleCell.Driver
delete() (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                               method), 293
                          method), 127
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.PneumaticSampleCell.Pneumat
                                                                                                                                                                                               method), 296
density (AFL.automation.prepare.Component.Component
                           property), 64, 396, 399
                                                                                                                                                                    deposit_obj() (AFL.automation.loading.PushPullSelectorSampleCell.Di
density (AFL.automation.prepare.Solute property), 389
                                                                                                                                                                                               method), 308
density (AFL.automation.prepare.Solvent property),
                                                                                                                                                                   deposit_obj() (AFL.automation.loading.PushPullSelectorSampleCell.Pu
                           393
                                                                                                                                                                                               method), 312
deposit_obj() (AFL.automation.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APISer
```

*method*), 319

method), 323

deposit\_obj() (AFL.automation.APIServer.Client.Client deposit\_obj() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX

 ${\tt deposit\_obj()} \ (AFL. automation. APIS erver. Driver. Driver {\tt deposit\_obj()} \ (AFL. automation. loading. Two Selector Blowout Sample Celling and the property of the$ 

method), 19, 100, 158

method), 21, 162, 165

| method), 365   | attribute), 604   |
|--|---|
| ${\tt deposit\_obj()} \ (AFL. automation. loading. Two Selector Blow {\tt dessinp} )$  |   |
| method), 370   | attribute), 612   |
| ${\tt deposit\_obj()} \ (AFL. automation. prepare. Deck Builder Widges {\tt Chirp}$  |   |
| method), 419   | attribute), 633   |
| deposit_obj() (AFL.automation.prepare.Dummy_OT2_DdiescrIm)   |   |
| method), 474   | attribute), 653   |
| deposit_obj() (AFL.automation.prepare.Dummy_OT2_Ddisscripp method), 477  | httyn_(All_Lantvanation.prepare.SweepBuilderWidget.Checkbox attribute), 662 |
| deposit_obj()(AFL.automation.prepare.OT2Client.Cliendescrip  |   |
| method), 482   | attribute), 678   |
| deposit_obj() (AFL.automation.prepare.OT2Client.OT2@lescrip  |   |
| method), 485   | attribute), 698   |
| deposit_obj() (AFL.automation.prepare.SampleSeriesWidesactlips   | ntrion_allow_html   |
| method), 585   | (AFL.automation.prepare.DeckBuilderWidget.Checkbox                          |
| ${\tt deposit\_obj()} \ (AFL. automation. sample. Casting Server. Casting Server$  | vattribute), 411  |
|  | tion_allow_html   |
| ${\tt deposit\_obj()} \ (AFL. automation. sample. Casting Server. Client$  | (AFL.automation.prepare.DeckBuilderWidget.Dropdown                          |
| method), 724   | attribute), 424   |
| deposit_obj() (AFL.automation.sample.CastingServer.Ddescrip  |   |
| method), 726   | (AFL.automation.prepare.DeckBuilderWidget.Label                             |
| deposit_obj() (AFL.automation.sample.CastingServer.OT2Client   |   |
| <pre>method), 729 descrip deposit_obj() (AFL.automation.sample_env.TemperatureDeck.Dri</pre>   | tion_allow_html   |
| method), 733   | attribute), 458   |
| deposit_obj() (AFL.automation.sample_env.Temperaturedesktflep  |   |
| method), 735   | (AFL.automation.prepare.PrepareWidget.Checkbox                              |
| description (AFL.automation.prepare.DeckBuilderWidget.Button   |   |
|  | tion_allow_html   |
| ${\tt description} (AFL. automation. prepare. Deck Builder Widget. Check because of the property of the propert$   |   |
| attribute), 411  | attribute), 513   |
| ${\tt description} (AFL. automation. prepare. Deck Builder Widge {\tt description})$   | ntrivon_allow_html  |
| attribute), 424  | (AFL. automation. prepare. Prepare Widget. Label                            |
| ${\tt description} (AFL. automation. prepare. Deck Builder Widget. Label$  |   |
|  | tion_allow_html   |
|  | (AFL.automation.prepare.PrepareWidget.Text                                  |
| attribute), 458  | attribute), 552   |
| description (AFL.automation.prepare.PrepareWidget.Butdescrip   |   |
| attribute), 494 description (AFL.automation.prepare.PrepareWidget.Checkbox   | (AFL.automation.prepare.SampleSeriesWidget.Checkbox attribute), 577         |
| - 1 1 1  | tion_allow_html   |
| description (AFL.automation.prepare.PrepareWidget.Dropdown   |   |
| attribute), 513  | attribute), 588   |
| description (AFL.automation.prepare.PrepareWidget.Labbescrip   |   |
| attribute), 529  | (AFL.automation.prepare.SampleSeriesWidget.IntText                          |
| description(AFL.automation.prepare.PrepareWidget.Text  | attribute), 604   |
| attribute), 552 descrip  | tion_allow_html   |
| ${\tt description} (AFL. automation. prepare. Sample Series Widget. Button$  | (AFL. automation. prepare. Sample Series Widget. Label                      |
| attribute), 568  | attribute), 612   |
| ${\tt description} (AFL. automation. prepare. Sample Series Widgetes {\tt Maxikp} in {\tt Maximum} in$ |   |
| attribute), 577  | (AFL.automation.prepare.SampleSeriesWidget.Text                             |
| description (AFL.automation.prepare.SampleSeriesWidget.FloatTe   |   |
|  | tion_allow_html   |
| ${\tt description} (AFL. automation. prepare. Sample Series Widget. Int Text$  | (AFL.automation.prepare.SweepBuilderWidget.Checkbox                         |

| attribute), 662  |                                 | property), 698                          |   |                     |
|--|---------------------------------|---|---|---------------------|
| description_allow_html   | deseria                         | lize()                                  | (in                                     | module              |
| (AFL. automation. prepare. Sweep Builder Widget. L   | abel                            | AFL.automation.sha                      | ıred.serialization),                    | 86,                 |
| attribute), 678  |                                 | 811                                     |   |                     |
| description_allow_html   | Diffrac                         | tionLabeler                             | (class                                  | in                  |
| (AFL.automation.prepare.SweepBuilderWidget.To attribute), 698  | ext                             | <i>AFL.automation.sha</i> 81, 759, 768  | ıred.DiffractionLab                     | peler),             |
| description_tooltip  | Diffrac                         | tionLabelerModel                        | . (class                                | in                  |
| (AFL.automation.prepare.DeckBuilderWidget.Ch<br>property), 411   | eckbox                          | <i>AFL.automation.sha</i> 81, 761, 768  | ıred.DiffractionLab                     | peler),             |
| description_tooltip  | Diffrac                         | tionLabelerView                         | (class                                  | in                  |
| (AFL.automation.prepare.DeckBuilderWidget.Dr   | opdown                          | AFL.automation.sha                      | ired.DiffractionLab                     | peler),             |
| property), 424   |                                 | 81, 761, 769                            |   |                     |
| description_tooltip  | Digital                         | OutPressureContr                        | roller (class                           | in                  |
| (AFL. automation. prepare. Deck Builder Widget. La   | bel                             | AFL.automation.loa                      | ding.DigitalOutPro                      | essureController),  |
| property), 440   |                                 | 36, 234, 236                            |   |                     |
| description_tooltip  | disable                         | ed (AFL.automation.p                    | repare.DeckBuilde                       | rWidget.Button      |
| (AFL.automation.prepare.DeckBuilderWidget.Tex  | xt                              | attribute), 402                         |   |                     |
| property), 458   | disable                         | ed (AFL.automation.p                    | repare.DeckBuilde                       | rWidget.Checkbox    |
| description_tooltip  |                                 | attribute), 411                         |   |                     |
| (AFL. automation. prepare. Prepare Widget. Checkber Midget. Checkber Mid | adisable                        | ed (AFL.automation.p                    | repare.DeckBuilde                       | rWidget.Dropdown    |
| property), 503   |                                 | attribute), 424                         |   |                     |
| description_tooltip  | disable                         | ed (AFL.automation.p                    | repare.DeckBuilde                       | rWidget.Text        |
| (AFL. automation. prepare. Prepare Widget. Dropdomation and the properties of the  | own                             | attribute), 457                         |   |                     |
| property), 513   | disable                         | ed (AFL.automation.p                    | repare.PrepareWid                       | lget.Button         |
| description_tooltip  |                                 | attribute), 494                         |   |                     |
| (AFL.automation.prepare.PrepareWidget.Label property), 529   | disable                         | ed (AFL.automation.p<br>attribute), 503 | repare.PrepareWid                       | lget.Checkbox       |
| description_tooltip  | disable                         | ed (AFL.automation.p.                   | repare.PrepareWid                       | lget.Dropdown       |
| (AFL.automation.prepare.PrepareWidget.Text   |                                 | attribute), 513                         |   |                     |
| property), 552   | disable                         | ed (AFL.automation.p                    | orepare.PrepareWi                       | dget.Text           |
| description_tooltip  |                                 | attribute), 551                         |   |                     |
| (AFL.automation.prepare.SampleSeriesWidget.C. property), 577   | <i>he</i> lai <i>ksla</i> ebole | ed (AFL.automation.p<br>attribute), 568 | repare.SampleSerie                      | esWidget.Button     |
| description_tooltip  | disable                         | ed (AFL.automation.p                    | repare.SampleSerie                      | esWidget.Checkbox   |
| (AFL.automation.prepare.SampleSeriesWidget.F.  |                                 |   | · F · · · · · · · · · · · · · · · · · · |                     |
| property), 588   |                                 | ed (AFL.automation.p.                   | repare.SampleSeri                       | esWidget.FloatText  |
| description_tooltip  |                                 | attribute), 587                         |   | O                   |
| (AFL.automation.prepare.SampleSeriesWidget.In  | <i>ti</i> dä <i>xs</i> rable    |   | repare.SampleSeri                       | esWidget.IntText    |
| property), 604   |                                 | attribute), 603                         |   |                     |
| description_tooltip  | disable                         | ed (AFL.automation.p                    | repare.SampleSeri                       | esWidget.Text       |
| (AFL.automation.prepare.SampleSeriesWidget.Lo  |                                 | attribute), 632                         |   |                     |
| property), 612   | disable                         | ed (AFL.automation.p                    | repare.SweepBuild                       | erWidget.Button     |
| description_tooltip  |                                 | attribute), 653                         |   |                     |
| (AFL. automation. prepare. Sample Series Widget. Teaching the series with the series of the series with the  | ex <b>d</b> isable              | ed (AFL.automation.p                    | repare.SweepBuild                       | erWidget.Checkbox   |
| property), 633   |                                 | attribute), 662                         |   |                     |
| description_tooltip  | disable                         | ed (AFL.automation.p                    | repare.SweepBuild                       | erWidget.Text       |
| (AFL. automation. prepare. Sweep Builder Widget. Comparison of the property  | heckbox                         | attribute), 697                         |   |                     |
| property), 662   | discove                         | r_server_by_name                        | : ()                                    |                     |
| description_tooltip  |                                 | (AFL.automation.AF                      | PIServer.Client.Ser                     | verDiscovery        |
| (AFL. automation. prepare. Sweep Builder Widget. Line was a substitution of the property of  | abel                            | method), 163                            |   |                     |
| property), 678   | discove                         | r_server_by_name                        |   |                     |
| description_tooltip  |                                 | (AFL.automation.sh                      |   | ery.ServerDiscovery |
| (AFL.automation.prepare.SweepBuilderWidget.To  | ext                             | method), 85, 792, 80                    | )9                                      |                     |

```
dispatch_request() (AFL.automation.APIServer.APISerdaysEtusetc() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 123
                                                                                               method), 800
dispense() (AFL.automation.loading.ChemyxSyringePumdrG\text{2}povixSyringePuffpautomation.APIServer.APIServer.ServiceInfo
             method), 35, 231, 233
                                                                                               method), 150
dispense() (AFL.automation.loading.ChemyxSyringePumptsyringeRump() (AFL.automation.shared.ServerDiscovery.AsyncServiceIn
             method), 232
                                                                                               method), 782
dispense() (AFL.automation.loading.DummyPump.DummdrSuppointer() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 38, 241, 243
                                                                                               method), 800
dispense() (AFL.automation.loading.DummyPump.SyringhtSurgervice() (AFL.automation.APIServer.APIServer.ServiceInfo
             method), 242
                                                                                               method), 151
dispense() (AFL.automation.loading.NE1kSyringePump.NEikSpringePump.AFL.automation.shared.ServerDiscovery.AsyncServiceIn
             method), 41, 265, 267
                                                                                               method), 783
dispense() (AFL.automation.loading.NE1kSyringePump.Siminge() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 266
                                                                                                method), 801
dispense() (AFL:automation.loading.PressureControllerAckPsurpePressureUntoHartAckPutribServer.APIServer.ServiceInfo
             method), 46, 303, 305
                                                                                               method), 151
dispense() (AFL.automation.loading.PressureControllerAthSurteStyli) (AFLinquitomation.shared.ServerDiscovery.AsyncServiceInfo
             method), 305
                                                                                               method), 783
dispense() (AFL.automation.loading.SyringePump.SyringeRsumpext() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 58, 361
                                                                                               method), 801
dispense_rate() (AFL.automation.prepare.OT2Client.OTBC/hieratrdown_appcontext()
             method), 67, 485, 487
                                                                                               (AFL.automation.APIServer.APIServer.Flask
dispense_rate() (AFL.automation.sample.CastingServer.OT2Clienmethod), 133
                                                                                  do_teardown_request()
             method), 729
dispenseRunning() (AFL.automation.loading.DigitalOutPressureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHees
             method), 235
                                                                                               method), 132
dispenseRunning() (AFL.automation.loading.DigitalOutRoassu(AAClantrollbeautronsshureClosservkleDiscovery.AsyncServiceBrowser
             method), 236
                                                                                               attribute), 779
dispenseRunning() (AFL.automation.loading.PressureColored & HELasstone attornshared.ServerDiscovery.ServiceBrowser
             method), 46, 302
                                                                                               attribute), 794
dispenseRunning() (AFL.automation.loading.UltimusVPPcoxbleeVivatiMArtRposSeeAGoutoaller
             method), 375
                                                                                               AFL.automation.loading.DoubleViciMultiposSelector),
dispenseRunning() (AFL.automation.loading.UltimusVPressureController,UltimusVPressureController
             method), 377
                                                                                  draw_deck() (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderW
display (AFL.automation.prepare.DeckBuilderWidget.Layout
                                                                                               method), 65, 421, 471
                                                                                  draw_peaks() (AFL.automation.shared.DataLabelerWidget.DataLabelerV
             attribute), 448
display (AFL.automation.prepare.PrepareWidget.Layout
                                                                                               method), 77, 740, 748
             attribute), 537
                                                                                  draw_peaks() (AFL.automation.shared.DiffractionLabeler.DiffractionLab
display (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                                                               method), 81, 760, 768
                                                                                  {\tt drive}\, (AFL. automation. instrument. Seabreeze UVV is. Path
             attribute), 620
display (AFL.automation.prepare.SweepBuilderWidget.Layout
                                                                                               property), 213
             attribute), 686
                                                                                  Driver (class in AFL.automation.APIServer.Driver), 21,
display()
                                        (in
                                                                    module
                                                                                                166, 171
             AFL.automation.shared.widgetui), 814
                                                                                  Driver (class in AFL.automation.APIServer.DummyDriver),
dns_addresses() (AFL.automation.APIServer.APIServer.ServiceInf@73
             method), 150
                                                                                  Driver (class in AFL.automation.APIServer.DummyOT2Driver),
dns_addresses() (AFL.automation.shared.ServerDiscovery.AsyncSetNiceInfo
                                                                                  Driver (class in AFL.automation.instrument.DummySAS),
             method), 782
dns_addresses() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 800
                                                                                  Driver (class in AFL.automation.instrument.I22SAXS),
dns_nsec() (AFL.automation.APIServer.APIServer.ServiceInfo
                                                                                  Driver (class in AFL.automation.instrument.SeabreezeUVVis),
             method), 150
dns_nsec() (AFL.automation.shared.ServerDiscovery.AsyncServiceInf6
             method), 782
                                                                                  Driver (class in AFL.automation.instrument.SpecScreen Driver),
```

```
219
                                                                                                                                         AFL.automation.prepare.Dummy_OT2_Driver),
Driver (class in AFL.automation.loading.LoadStopperDriver),
                                                                                                                                         66, 475, 478
                                                                                                                     dummy_reset_tank_levels()
Driver (class in AFL.automation.loading.OneSelectorBlowoutSample(AATI),automation.APIServer.DummyDriver.DummyDriver
                                                                                                                                         method), 23, 178, 180
Driver (class in AFL.automation.loading.PneumaticPressurdsimmplrestlt_tank_levels()
                                                                                                                                         (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
                                                                                                                                         method), 24, 184, 186
Driver (class in AFL.automation.loading.PneumaticSampleCell),
                                                                                                                     DummvDriver
                                                                                                                                                                                     (class
                                                                                                                                                                                                                                  in
Driver (class in AFL.automation.loading.PushPullSelectorSampleCelA)FL.automation.APIServer.DummyDriver),
                                                                                                                                         23, 176, 179
Driver (class in AFL.automation.loading.RSoXSSolutionSabundarQbbijver
                                                                                                                                                                                     (class
                                                                                                                                         AFL.automation.APIServer.DummyOT2Driver),
Driver (class in AFL.automation.loading.TwoSelectorBlowoutSample@ell\)\\\83, 186
                                                                                                                     DummyPump (class in AFL.automation.loading.DummyPump),
Driver (class in AFL.automation.prepare.Dummy_OT2_Driver),
                                                                                                                                         38, 240, 242
                                                                                                                     DummySAS (class in AFL.automation.instrument.DummySAS),
                                                                                                                                         28, 196, 198
Driver (class in AFL. automation. sample. Casting Server),
                                                                                                                     DummySensor1
                                                                                                                                                                                     (class
                                                                                                                                                                                                                                  in
Driver (class in AFL.automation.sample env.TemperatureDeck),
                                                                                                                                         AFL.automation.loading.Sensor),
                                                                                                                                                                                                                              333,
                                                                                                                                         339
driver_status() (AFL.automation.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIS
                                                                                                                                                                                      (class
                                                                                                                                                                                                                                  in
                   method), 18, 100, 158
                                                                                                                                         AFL.automation.loading.Sensor),
                                                                                                                                                                                                                             336,
driver_status() (AFL.automation.APIServer.Client.Client
                   method), 20, 161, 164
driver_status()(AFL.automation.loading.LoadStopperDriver.Client
                   method), 246
                                                                                                                     emit() (AFL.automation.APIServer.APIServer.FileHandler
driver_status()(AFL.automation.loading.Sensor.DummySensor2 method), 104
                   method), 53, 337, 340
                                                                                                                     emit() (AFL.automation.APIServer.APIServer.SMTPHandler
driver_status() (AFL.automation.prepare.DeckBuilderWidget.Clientthod), 146
                   method), 418
                                                                                                                      emit() (AFL.automation.prepare.Component.Component
driver_status() (AFL.automation.prepare.OT2Client.Client
                                                                                                                                         method), 64, 396, 398
                   method), 481
                                                                                                                     emit() (AFL.automation.prepare.Solute method), 389
driver_status() (AFL.automation.prepare.OT2Client.OT&AClient (AFL.automation.prepare.Solvent method), 393
                   method), 485
                                                                                                                     emit_protocol() (AFL.automation.prepare.OT2Client.PipetteAction
driver_status() (AFL.automation.prepare.SampleSeriesWidget.Climethod), 487
                   method), 584
                                                                                                                     emit\_protocol() (AFL.automation.prepare.PipetteAction
driver_status() (AFL.automation.sample.CastingServer.Client method), 387
                   method), 723
                                                                                                                     emit_protocol()
                                                                                                                                                                    (AFL.automation.prepare.Sample
driver_status() (AFL.automation.sample.CastingServer.OT2Clienmethod), 387
                   method), 729
                                                                                                                     Empty, 770
Dropdown (class in AFL.automation.prepare.DeckBuilderWiehnty) () (AFL.automation.APIServer.APIServer.MutableQueue
                                                                                                                                         method), 141
Dropdown (class in AFL.automation.prepare.PrepareWidgetempty() (AFL.automation.shared.MutableQueue.MutableQueue
                                                                                                                                         method), 82, 770, 771
drySyringe() (AFL.automation.loading.OneSelectorBlow@mfStyRic@phi.@meSelectorBlowoutSampleCell
                   method), 42, 272, 281
                                                                                                                     emptySyringe() (AFL.automation.loading.ChemyxSyringePump.Chemyx
drySyringe() (AFL.automation.loading.OneSelectorBlowoutSampleGell.TwpSelectorBlowoutSampleCell
                   method), 277
                                                                                                                     emptySyringe() (AFL.automation.loading.ChemyxSyringePump.SyringeF
drySyringe() (AFL.automation.loading.RSoXSSolutionSampleCell.RSaKSSplWiDnSampleCell
                   method), 50, 322, 328
                                                                                                                     emptySyringe() (AFL.automation.loading.DummyPump.DummyPump
\label{lower} \verb|drySyringe()| (AFL. automation. loading. Two Selector Blowout Sample \textit{GethTuto}. Selector Blowout Sample \textit{Cell}. Two Selector Blowout Sample Blowout 
                   method), 60, 370, 374
                                                                                                                     emptySyringe() (AFL.automation.loading.DummyPump.SyringePump
Dummy_OT2_Driver
                                                                     (class
                                                                                                             in
                                                                                                                                         method), 242
```

emptySyringe() (AFL.automation.loading.NE1kSyringePump.NE1kSpetImpeP,นีกปฏ

```
method), 41, 265, 267
                                                                                                                                                                    ensure_sync() (AFL.automation.APIServer.APIServer.Flask
emptySyringe() (AFL.automation.loading.NE1kSyringePump.SyringeePlhod), 124
                          method), 266
                                                                                                                                                                    Enum (class in AFL.automation.prepare.PrepType), 488
emptySyringe() (AFL.automation.loading.PressureControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpultonesstireControllerAsPhilmpu
                          method), 46, 303, 305
                                                                                                                                                                                               erty), 117
emptySyringe() (AFL.automation.loading.PressureControlleriAssOmmen.SyriagalRoller
                                                                                                                                                                                  FLASK_DEBUG, 118
                           method), 305
emptySyringe() (AFL.automation.loading.SyringePump.SpaingAFL.automation.instrument.SeabreezeUVVis),
                          method), 58, 361
                                                                                                                                                                                                208
\verb|encode()| (AFL. automation. instrument. Seabreeze UVV is. Eqerror\_handler\_spec(AFL. automation. APIS erver. APIS erver. Flask) | (AFL. automation. instrument. Seabreeze UVV is. Eqerror\_handler\_spec(AFL. automation. APIS erver. APIS erver. Flask) | (AFL. automation. APIS erver. 
                           method), 208
                                                                                                                                                                                                attribute), 131
                                                                                                                                                                    errorhandler() (AFL.automation.APIServer.APIServer.Flask
encode_key_loader()
                          (AFL.automation.APIServer.APIServer.JWTManager
                                                                                                                                                                                                method), 127
                                                                                                                                                                     example_label_cb() (AFL.automation.prepare.PrepareWidget.SampleSe
                          method), 138
\verb"endpoint"()" (AFL. automation. APIS erver. APIS erver. Flask
                                                                                                                                                                                                method), 547
                                                                                                                                                                    example_label_cb() (AFL.automation.prepare.SampleSeriesWidget.Sam
                          method), 127
enforce_units()
                                                                                                                                         module
                                                                                                                                                                                               method), 70, 629, 646
                                                                                            (in
                          AFL.automation.prepare.Component), 395
                                                                                                                                                                                                            (AFL.automation.APIServer.Driver.Driver
                                                                                                                                                                    execute()
enforce_units()
                                                                                            (in
                                                                                                                                          module
                                                                                                                                                                                               method), 22, 168, 172
                          AFL.automation.shared.units), 87, 812
                                                                                                                                                                    execute() (AFL.automation.APIServer.DummyDriver.Driver
enqueue() (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                                                                                               method), 175
                          method), 19, 100, 158
                                                                                                                                                                    execute() (AFL.automation.APIServer.DummyDriver.DummyDriver
                                           (AFL.automation.APIServer.Client.Client
enqueue()
                                                                                                                                                                                                method), 178
                          method), 20, 161, 164
                                                                                                                                                                    execute() (AFL.automation.APIServer.DummyOT2Driver.Driver
enqueue() (AFL.automation.loading.LoadStopperDriver.Client
                                                                                                                                                                                               method), 182
                           method), 246
                                                                                                                                                                     execute() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
enqueue() (AFL.automation.prepare.DeckBuilderWidget.Client
                                                                                                                                                                                                method), 24, 184, 186
                          method), 418
                                                                                                                                                                     execute() (AFL.automation.instrument.DummySAS.Driver
enqueue() (AFL.automation.prepare.OT2Client.Client
                                                                                                                                                                                                method), 195
                           method), 482
                                                                                                                                                                    execute() (AFL.automation.instrument.DummySAS.DummySAS
enqueue() (AFL.automation.prepare.OT2Client.OT2Client
                                                                                                                                                                                                method), 197
                          method), 485
                                                                                                                                                                    execute() (AFL.automation.instrument.I22SAXS.Driver
enqueue() (AFL.automation.prepare.SampleSeriesWidget.Client
                                                                                                                                                                                                method), 201
                           method), 584
                                                                                                                                                                     execute() (AFL.automation.instrument.I22SAXS.I22SAXS
enqueue() (AFL.automation.sample.CastingServer.Client
                                                                                                                                                                                                method), 203
                          method), 723
                                                                                                                                                                    execute() (AFL.automation.instrument.SeabreezeUVVis.Driver
enqueue() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                                                                                                                method), 207
                           method), 729
                                                                                                                                                                    execute() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis
enqueued_base() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                method), 217
                          method), 20, 161, 164
                                                                                                                                                                    execute() (AFL.automation.instrument.SpecScreen Driver.Driver
enqueued_base() (AFL.automation.loading.LoadStopperDriver.Cliemethod), 221
                                                                                                                                                                    execute() (AFL.automation.instrument.SpecScreen_Driver.SpecScreen_D
                          method), 246
enqueued_base() (AFL.automation.prepare.DeckBuilderWidget.Clienethod), 31, 222, 224
                                                                                                                                                                    execute() (AFL.automation.loading.LoadStopperDriver.Driver
                          method), 418
enqueued_base() (AFL.automation.prepare.OT2Client.Client
                                                                                                                                                                                                method), 248
                                                                                                                                                                    \verb"execute()" (AFL. automation. loading. Load Stopper Driver. Load Stop
                           method), 482
enqueued_base() (AFL.automation.prepare.OT2Client.OT2Client method), 251
                           method), 485
                                                                                                                                                                    execute() (AFL. automation. loading. One Selector Blowout Sample Cell. Drive
enqueued_base() (AFL.automation.prepare.SampleSeriesWidget.Climathod), 269
                                                                                                                                                                    {\tt execute()} \ (AFL. automation. loading. One Selector Blowout Sample Cell. One Selector Blow out Sample Cell. One Selector Blow One 
                          method), 584
enqueued_base() (AFL.automation.sample.CastingServer.Client
                                                                                                                                                                                            method), 273
                          method), 723
                                                                                                                                                                    execute() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Two
enqueued_base() (AFL.automation.sample.CastingServer.OT2Clienmethod), 277
```

```
execute() (AFL.automation.loading.PneumaticPressureSabgaleCrell_Daimes_in_(AFL.automation.shared.DiffractionLabeler.Ordinal.
             method), 283
                                                                                                 attribute), 763
execute() (AFL.automation.loading.PneumaticPressureSatripleCaddeRraequalussicalPAAsLuausSoumapliaGiebbstrument.FileCamera),
                                                                                                29, 198, 199
             method), 287
execute() (AFL.automation.loading.PneumaticSampleCelEDreHandler
                                                                                                                               (class
             method), 293
                                                                                                AFL.automation.APIServer.APIServer), 103
execute() (AFL.automation.loading.PneumaticSampleCelEPhtenn())(AFhphn())(AFhphn())
             method), 296
                                                                                                 method), 104
execute() (AFL.automation.loading.PushPullSelectorSamfile\textsq.dt.Dri\textsq.eFL.automation.APIServer.APIServer.SMTPHandler
             method), 308
                                                                                                method), 146
execute() (AFL.automation.loading.PushPullSelectorSamfileCelliPashRAMSelectorSainplACelServer.QueueDaemon.DataTrashcan
                                                                                                method), 189
             method), 312
execute() (AFL.automation.loading.RSoXSSolutionSamplaCondLibragrequest() (AFL.automation.APIServer.APIServer.Flask
                                                                                                 method), 124
             method), 319
execute() (AFL.automation.loading.RSoXSSolutionSampleCall_RSoXSSolutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmStatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstatutionmstat
              method), 323
                                                                                                 class method), 787
execute() (AFL.automation.loading.TwoSelectorBlowoutStimpde@eltyPeniylbry_name()
             method), 365
                                                                                                (AFL.automation.APIServer.Client.ServerDiscovery
execute() (AFL.automation.loading.TwoSelectorBlowoutSampleCellnTextorSel.getforBlowoutSampleCell
             method), 370
                                                                                   find_server_by_name()
execute() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
                                                                                                (AFL.automation.shared.ServerDiscovery.ServerDiscovery
             method), 474
                                                                                                method), 85, 792, 809
execute() (AFL.automation.prepare.Dummy_OT2_Driver.flimduse@Nier_Ibmivpartial_name()
                                                                                                (AFL.automation.APIServer.Client.ServerDiscovery
              method), 477
execute() (AFL.automation.sample.CastingServer.CastingServer
                                                                                                method), 164
             method), 720
                                                                                   find_server_by_partial_name()
execute() (AFL.automation.sample.CastingServer.Driver
                                                                                                 (AFL.automation.shared.ServerDiscovery.ServerDiscovery
             method), 726
                                                                                                method), 85, 793, 809
execute() (AFL.automation.sample_env.TemperatureDeckfDnideserver_by_property_match()
             method), 733
                                                                                                (AFL.automation.APIServer.Client.ServerDiscovery
execute() (AFL.automation.sample_env.TemperatureDeck.TemperatureDeck, 164
              method), 735
                                                                                   find_server_by_property_match()
exists() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                (AFL.automation.shared.ServerDiscovery.ServerDiscovery
                                                                                                method), 85, 793, 809
             method), 213
expanduser() (AFL.automation.instrument.SeabreezeUVVfixPath(AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
             method), 213
                                                                                                method), 745
expired_token_loader()
                                                                                   fit() (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
             (AFL.automation.APIServer.APIServer.JWTManager
                                                                                                method), 766
             method), 138
                                                                                   fit_transform() (AFL.automation.shared.DataLabelerWidget.OrdinalEnd
expose() (AFL.automation.instrument.DummySAS.DummySAS
                                                                                                method), 745
             method), 28, 197, 198
                                                                                   fit_transform() (AFL.automation.shared.DiffractionLabeler.OrdinalEn
expose() (AFL.automation.instrument.I22SAXS.I22SAXS
                                                                                                method), 766
             method), 29, 203, 204
                                                                                  Flask (class in AFL.automation.APIServer.APIServer),
{\tt extensions} \, (AFL. automation. APIS erver. APIS erver. Flask
                                                                                                 105
                                                                                  FLASK_DEBUG, 118
             attribute), 114
extract_var() (AFL.automation.shared.DatasetWidget.DatberuWidgetutomation.prepare.DeckBuilderWidget.Layout
             method), 79, 751, 757
                                                                                                attribute), 448
extract_var() (AFL.automation.shared.DatasetWidget.DathautWidgetLMuttahation.prepare.PrepareWidget.Layout
             method), 80, 753, 758
                                                                                                 attribute), 537
                                                                                   {\tt flex}\,(AFL. automation. prepare. Sample Series Widget. Layout
F
                                                                                                 attribute), 620
feature_names_in_(AFL.automation.shared.DataLabelerWadsA.Ordinton_pation_prepare.SweepBuilderWidget.Layout
                                                                                                 attribute), 686
             attribute), 742
                                                                                   flex_flow (AFL.automation.prepare.DeckBuilderWidget.Layout
```

| attribute), 448 method), 570   |     |
|--|-----|
| $\verb flex_flow  (AFL. automation. prepare. Prepare Widget. Layou \verb focus()  (AFL. automation. prepare. Sample Series Widget. Checkbox   Checkbox$ |     |
| attribute), 537 method), 578   |     |
| flex_flow(AFL.automation.prepare.SampleSeriesWidget.Hoogaus() (AFL.automation.prepare.SampleSeriesWidget.FloatText   |     |
| attribute), 620 method), 589   |     |
| flex_flow(AFL.automation.prepare.SweepBuilderWidget. <b>Eoyws</b> () (AFL.automation.prepare.SampleSeriesWidget.HBox   |     |
| attribute), 686 method), 596   |     |
| FloatText (class in AFL.automation.prepare.SampleSeries <b>Wodyss</b> ) (AFL.automation.prepare.SampleSeriesWidget.IntText method), 604  |     |
| FlowSelector (class in focus() (AFL.automation.prepare.SampleSeriesWidget.Label  |     |
| AFL.automation.loading.CetoniMultiPosValve), method), 612  |     |
| focus() (AFL.automation.prepare.SampleSeriesWidget.Text  |     |
| FlowSelector (class in method), 634  |     |
| AFL.automation.loading.DoubleViciMultiposSelector) (AFL.automation.prepare.SampleSeriesWidget.VBox   |     |
| 238 method), 641   |     |
| FlowSelector (class in focus() (AFL.automation.prepare.SweepBuilderWidget.Button   |     |
| AFL.automation.loading.FlowSelector), method), 654   |     |
| 38, 243 focus() (AFL.automation.prepare.SweepBuilderWidget.Checkbox  |     |
| FlowSelector (class in method), 662  |     |
| AFL.automation.loading.ViciMultiposSelector), focus() (AFL.automation.prepare.SweepBuilderWidget.HBox  |     |
| 378 <i>method</i> ), 670   |     |
| flush() (AFL.automation.APIServer.APIServer.FileHandlefocus() (AFL.automation.prepare.SweepBuilderWidget.Label   |     |
| method), 104 method), 678  |     |
| flush() (AFL.automation.APIServer.APIServer.SMTPHandbecus() (AFL.automation.prepare.SweepBuilderWidget.Text  |     |
| method), 146 method), 699  |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.Buttbocus() (AFL.automation.prepare.SweepBuilderWidget.VBox  |     |
| method), 403 method), 706 focus() (AFL.automation.prepare.DeckBuilderWidget.Che <b>Ekhan</b> at() (AFL.automation.APIServer.APIServer.FileHandler  |     |
| method), 411  method), 104   |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.Droftdormat() (AFL.automation.APIServer.APIServer.SMTPHandler  |     |
| method), 424  method), 146   |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.HBofformula (AFL.automation.prepare.Component.Component  |     |
| method), 432 property), 64, 396, 399   |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.Labetormula (AFL.automation.prepare.Solute property), 389  |     |
| method), 440 formula (AFL.automation.prepare.Solvent property),  |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.Text 393   |     |
| method), 459 from_dict() (AFL.automation.prepare.factory.Solution  |     |
| focus() (AFL.automation.prepare.DeckBuilderWidget.VBox class method), 715  |     |
| method), 466 from_dict() (AFL.automation.prepare.Solution class  |     |
| focus() (AFL.automation.prepare.PrepareWidget.Button method), 391  |     |
| method), 496 from_server_name() (AFL.automation.APIServer.Client.Client  |     |
| focus() (AFL.automation.prepare.PrepareWidget.Checkbox class method), 20, 161, 164   |     |
| method), 504 from_server_name() (AFL.automation.loading.LoadStopperDriver.Clie   | en  |
| focus() (AFL.automation.prepare.PrepareWidget.Dropdown class method), 245  |     |
| method), 513 from_server_name() (AFL.automation.prepare.DeckBuilderWidget.Clic   | er  |
| focus() (AFL.automation.prepare.PrepareWidget.HBox class method), 418  |     |
| method), 521 from_server_name() (AFL.automation.prepare.OT2Client.Client   |     |
| focus() (AFL.automation.prepare.PrepareWidget.Label class method), 481   |     |
| method), 529 from_server_name() (AFL.automation.prepare.OT2Client.OT2Client  |     |
| focus() (AFL.automation.prepare.PrepareWidget.Text class method), 485  | , . |
| method), 553 from_server_name() (AFL.automation.prepare.SampleSeriesWidget.Cl  | ıe  |
| focus() (AFL automation.prepare.PrepareWidget.VBox class method), 584  |     |
| method), 560 from_server_name() (AFL automation.sample.CastingServer.Client  |     |
| focus() (AFL.automation.prepare.SampleSeriesWidget.Button class method), 723   |     |

```
from_server_name() (AFL.automation.sample.CastingSepartQF2dlifatilts() (AFL.automation.loading.OneSelectorBlowoutSample
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 273
                                                          class method), 729
fromkeys() (AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.loading.OneSelectorBlowoutSanparCell:fathfullsdig(AFL.automation.load
                                                        method), 280
                                                                                                                                                                                                                                                                                                                                                                                                                       class method), 278
 fromkeys() (AFL.automation.loading.PneumaticPressureSampleCelledialullicsic) (AFL.automation.loading.PneumaticPressureSample
                                                        method), 289
                                                                                                                                                                                                                                                                                                                                                                                                                       class method), 282
fromkeys() (AFL.automation.loading.PneumaticSampleCalatlefem1tdicfaults() (AFL.automation.loading.PneumaticPressureSample
                                                          method), 299
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 287
fromkeys() (AFL.automation.loading.PushPullSelectorSargptAGedLdeftawtHtss() (AFL.automation.loading.PneumaticSampleCell.Dri
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 293
                                                          method), 315
{\tt fromkeys()} \ (AFL. automation. loading. RSoXSS olution Samp{\tt decChlerle declarite} \texttt{t.s.()} \ (AFL. automation. loading. Pneumatic Sample Cell. Pneumatic Sample Sample Cell. Pneumatic Sample Cell. Pneumatic Sample Sample Cell. Pneumatic Sample Sample Sample Sample Sample Sample Sample Samp
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 296
                                                          method), 326
fromkeys() (AFL.automation.loading.TwoSelectorBlowout\( \)arthorout\( \)
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 307
                                                          method), 372
fromkeys() (AFL.automation.shared.DatasetWidget.defaulgdither_defaults() (AFL.automation.loading.PushPullSelectorSampleC
                                                          method), 756
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 312
                                                                                                                                                                                                                                                                                                                                                            gather_defaults() (AFL.automation.loading.RSoXSSolutionSampleCell
Full, 771
 full_dispatch_request()
                                                                                                                                                                                                                                                                                                                                                                                                                       class method), 319
                                                           (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                                                                                            gather_defaults() (AFL.automation.loading.RSoXSSolutionSampleCell
                                                        method), 123
                                                                                                                                                                                                                                                                                                                                                                                                                       class method), 323
                                                                                                                                                                                                                                                                                                                                                             gather_defaults() (AFL.automation.loading.TwoSelectorBlowoutSample
G
                                                                                                                                                                                                                                                                                                                                                                                                                       class method), 364
\verb|gather_defaults()| (AFL. automation. APIS erver. Diver. Deather_defaults()| (AFL. automation. loading. Two Selector Blowout Sample of the Selector Blow
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 370
                                                          class method), 21, 167, 171
\verb|gather_defaults()| (AFL. automation. APIS erver. Dummy | \verb|gather_defaults()| (AFL. automation. prepare. Dummy | OT2 | Driver. Dri
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 473
                                                          class method), 174
gather_defaults()(AFL.automation.APIServer.DummyIDatherDudgfaplts()(AFL.automation.prepare.Dummy_OT2_Driver.Dum
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 478
                                                          class method), 178
{\tt gather\_defaults()} \ (AFL. automation. APIS erver. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Casting Server. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Dummy {\tt GFL} {\tt prodefaults()} \ (AFL. automation. sample. Dummy {\tt prodefaults()} \ (AFL. automation. sample. Sample. sample. Sample. Sample. sample. Sample
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 720
                                                           class method), 182
gather_defaults()(AFL.automation.APIServer.Dummy@F2DeficedefaultysCr)veAFL.automation.sample.CastingServer.Driver
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 725
                                                          class method), 185
 {\tt gather\_defaults()} \ (AFL. automation. instrument. Dummy {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. automation. sample\_env. Temperature Deck. Drive {\tt gather\_defaults()} \ (AFL. auto
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 732
                                                        class method), 194
{\tt gather\_defaults()} \ (AFL. automation. instrument. Dummy {\tt SASIPAL midgle SASI} {\tt ts()} \ (AFL. automation. sample\_env. Temperature Deck. Temperature
                                                                                                                                                                                                                                                                                                                                                                                                                        class method), 735
                                                          class method), 197
 gather\_defaults() (AFL.automation.instrument.I22SAXgenerate\_service\_broadcast()
                                                                                                                                                                                                                                                                                                                                                                                                                        (AFL.automation.APIServer.APIServer.Zeroconf
                                                          class method), 200
gather_defaults()(AFL.automation.instrument.I22SAXS.I22SAXSnethod), 156
                                                                                                                                                                                                                                                                                                                                                            generate_service_broadcast()
                                                          class method), 203
{\tt gather\_defaults()} \ (AFL. automation. instrument. Seabreeze UVV is. \textit{DAFF} automation. shared. Server Discovery. Zeroconfolia and the property of the p
                                                                                                                                                                                                                                                                                                                                                                                                                       method), 807
                                                          class method), 207
gather_defaults()(AFL.automation.instrument.Seabree9e0evest.Seabrevieevy()
                                                                                                                                                                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.APIServer.Zeroconf
                                                           class method), 217
gather_defaults() (AFL.automation.instrument.SpecScreen_Driventstand), 156
                                                                                                                                                                                                                                                                                                                                                             generate_service_query()
                                                          class method), 220
{\tt gather\_defaults()} \ (AFL. automation. instrument. Spec Screen\_Driver. Spec Superationation in the property of the content of the property of the propert
                                                                                                                                                                                                                                                                                                                                                                                                                       method), 807
                                                           class method), 223
{\tt gather\_defaults()} \ (AFL. automation. loading. Load Stopp {\tt approximate} {\tt prupregister\_all\_services()}
                                                                                                                                                                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.APIServer.Zeroconf
                                                          class method), 248
gather_defaults() (AFL.automation.loading.LoadStopperDriver.LWatts() perDriver
                                                                                                                                                                                                                                                                                                                                                             generate_unregister_all_services()
                                                           class method), 251
{\tt gather\_defaults()}\ (AFL. automation. loading. One Selector Blowout {\tt Samble Comption}, shared. Server Discovery. Zero configuration and the state of the s
                                                                                                                                                                                                                                                                                                                                                                                                                      method), 807
                                                          class method), 269
```

| get()      | (AFL.automation.APIServer.APIServer.Flask  |                             | method), 185   |
|------------|--|-----------------------------|--|
|            | method), 128   | get_con                     | fig() (AFL.automation.instrument.DummySAS.Driver   |
| get()(     | AFL. automation. APIS erver. APIS erver. Mutable Que   | ue                          | method), 194   |
|            | method), 142   | get_con                     | fig() (AFL.automation.instrument.DummySAS.DummySAS   |
| get()(     | AFL.automation.APIServer.Driver.PersistentConfig   | ,                           | method), 197   |
|            | method), 171   | get_con                     | fig() (AFL.automation.instrument.I22SAXS.Driver  |
| get()(     | AFL.automation.APIServer.QueueDaemon.DataTr  |                             | method), 200   |
|            | method), 189   |                             | fig() (AFL.automation.instrument.I22SAXS.I22SAXS   |
| get()(     | AFL.automation.loading.OneSelectorBlowoutSamp  | _                           |  |
| 5 ,        | method), 280   |                             | fig() (AFL.automation.instrument.SeabreezeUVVis.Driver   |
| aet()(     | AFL.automation.loading.PneumaticPressureSampl  |                             |  |
| 3()        | method), 289   |                             | fig() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUV  |
| get()(     | AFL.automation.loading.PneumaticSampleCell.dej   | _                           | method), 217   |
| gcc() (    | method), 299   |                             | fig() (AFL.automation.instrument.SpecScreen_Driver.Driver  |
| aet () (   | AFL.automation.loading.PushPullSelectorSampleC   |                             |  |
| get() (    | method), 315   |                             | fig() (AFL.automation.instrument.SpecScreen_Driver.SpecScree   |
| aa+()(     | memoa), 515<br>AFL.automation.loading.RSoXSSolutionSampleCea   |                             |  |
| get()(     |  |                             |  |
| ~a+()(     | method), 326   |                             | fig() (AFL.automation.loading.LoadStopperDriver.Client   |
| get()(     | AFL.automation.loading.TwoSelectorBlowoutSamp  |                             |  |
|            | method), 372   | get_con                     | fig() (AFL.automation.loading.LoadStopperDriver.Driver   |
| get()(     | AFL.automation.shared.DatasetWidget.defaultdict  |                             | method), 248   |
|            | method), 756   | -                           | fig() (AFL.automation.loading.LoadStopperDriver.LoadStopper  |
| get()(     | AFL.automation.shared. $M$ utable $Q$ ueue. $M$ utable $Q$ u   |                             | method), 251   |
|            | method), 82, 770, 771  | -                           | ${\tt fig()} \ (AFL. automation. loading. One Selector Blowout Sample Cell. \\$  |
| get()(     | AFL. automation. shared. Persistent Config. Mutable Number 1995. The properties of |                             | method), 269   |
|            | method), 772   |                             | $\verb fig()  (AFL. automation. loading. One Selector Blow out Sample Cell. \\$  |
| get()(     | AFL. automation. shared. Persistent Config. Persistent   |                             | method), 273   |
|            | method), 775   | get_con                     | $\verb fig()  (AFL. automation. loading. One Selector Blow out Sample Cell. \\$  |
| get_ad     | dress_and_nsec_records()   |                             | method), 278   |
|            | (AFL.automation.APIServer.APIServer.ServiceIn  | f <b>g</b> et_con           | ${\tt fig()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Description of the pressure of the pr$ |
|            | method), 151   |                             | method), 282   |
| get_ad     | dress_and_nsec_records()   | get_con                     | ${\tt fig()}\ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pressure Sample $ |
|            | (AFL.automation.shared.ServerDiscovery.Asynct  | ServiceInfo                 | omethod), 287  |
|            | method), 783   | get_con                     | fig() (AFL.automation.loading.PneumaticSampleCell.Driver   |
| get_ad     | dress_and_nsec_records()   |                             | method), 293   |
| _          |  | e <b>lge</b> to_con         | fig() (AFL.automation.loading.PneumaticSampleCell.Pneumatic  |
|            | method), 801   | - 0                         | method), 297   |
| get_co     |  | get_con                     | fig() (AFL.automation.loading.PushPullSelectorSampleCell.Dri   |
| <b>5</b> – | method), 384   | 5 –                         | method), 307   |
| aet co     |  | l exet.tDatom               | EtilyiageA_MLcahelomation.loading.PushPullSelectorSampleCell.Pus   |
| 900_00     | method), 80, 753, 758  |                             | method), 312   |
| get co     | 7  | ane#Wizha                   | refig() (AFL.automation.loading.RSoXSSolutionSampleCell.Drive  |
| gc c_co.   | method), 79, 751, 757  | ugeno <u>r</u> augu         | method), 319   |
| det co     |  | get con                     | fig() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX   |
| gcc_co.    | method), 20, 161, 164  | gc c_con                    | method), 323   |
| ant co     |  | act con                     | fig() (AFL.automation.loading.TwoSelectorBlowoutSampleCell   |
| get_co.    | method), 21, 167, 171  | get_con                     | method), 364   |
| aa+ aa     |  | Batuagon                    | fig() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.  |
| get_co.    |  | . <b>ig</b> en <u>e</u> kon |  |
|            | method), 174   | . D. + D                    | method), 370   |
| get_co     | - · · · · · · · · · · · · · · · · · · ·  | . We trutton                | (AFL.automation.prepare.DeckBuilderWidget.Client   |
|            | method), 178   |                             | method), 418   |
| get_co     |  | rgetLoon                    | fig() (AFL.automation.prepare.Dummy_OT2_Driver.Driver  |
|            | method), 182   |                             | method), 473   |
| get_co     | n±1g()(AFL.automation.APIServer.DummyOT2D  | rget.Laon                   | <b>ftiyD(r)</b> v@AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O  |

| method), 478   | method), 278   |
|--|--|
| $\verb"get_config()" (AFL. automation. prepare. OT 2Client. Client \verb"get_config") (AFL. automation. prepare. OT 2Client. Client. $   |  |
| method), 482   | method), 282   |
| $\verb"get_config" () \textit{ (AFL. automation. prepare. OT 2C lient. OT 2C lient. CT 2C lient) } and \textit{ (AFL. automation. prepare. OT 2C lient. OT 2C lient) } and \textit{ (AFL. automation. prepare. OT 2C lient. OT 2C lient) } and \textit{ (AFL. automation. prepare. OT 2C lient. OT 2C lient) } and \textit{ (AFL. automation. prepare. OT 2C lient. OT 2C lient) } and \textit{ (AFL. automation. prepare. OT 2C lient) } and  (AFL. automation. prepare. OT 2C li$ |  |
| method), 485   | method), 287   |
| $\verb"get_config" () \textit{ (AFL. automation. prepare. Sample Series Wid \textit{getChie})} \\$   |  |
| method), 584   | method), 293   |
| $\verb"get_config()" (AFL. automation. sample. Casting Server. Ca$   | ${f metrigs}$ () (AFL. automation. loading. Pneumatic Sample Cell. Pneumatics and the contraction of the c     |
| method), 720   | method), 297   |
| $\verb"get_config()" (AFL. automation. sample. Casting Server. Cliquet_config()) and the sample of the $   | ${\tt onfigs()}\ (AFL. automation. loading. Push Pull Selector Sample Cell. Date of the property of the propert$ |
| method), 723   | method), 307   |
| $get\_config()$ (AFL.automation.sample.CastingServer.Driget\_config()  | ${\tt onfigs()}\ (AFL. automation. loading. Push Pull Selector Sample Cell. $ |
| method), 725   | method), 312   |
| get_config() (AFL.automation.sample.CastingServer.OT2)@tieatc  | ${\tt onfigs()}\ (AFL. automation. loading. RSoXSS olution Sample Cell. Drive$   |
| method), 729   | method), 319   |
| <pre>get_config() (AFL.automation.sample_env.TemperatureDyet.Do</pre>  | intigs() (AFL.automation.loading.RSoXSSolutionSampleCell.RSo.  |
| method), 732   | method), 323   |
| get_config() (AFL.automation.sample_env.TemperatureDyet_Tea  | myfizigstyre DeFlL.automation.loading.TwoSelectorBlowoutSampleCeli   |
| method), 735   | method), 364   |
| <pre>get_configs() (AFL.automation.APIServer.Driver.Driverget_co</pre>   |  |
| method), 21, 167, 171  | method), 370   |
| get_configs() (AFL.automation.APIServer.DummyDrivergerivec   |  |
| method), 174   | method), 473   |
| get_configs() (AFL.automation.APIServer.DummyDrivergetunga   |  |
| method), 178   | method), 478   |
| get_configs() (AFL.automation.APIServer.DummyOT2Dgiver.Du  |  |
| method), 182   | method), 720   |
| get_configs() (AFL.automation.APIServer.DummyOT2Dgiver.Du  |  |
| method), 185   | method), 725   |
| get_configs() (AFL.automation.instrument.DummySAS.Dycite.co  |  |
| method), 194   | method), 732   |
| get_configs() (AFL.automation.instrument.DummySAS.Dyertnroc  |  |
| method), 197   | method), 735   |
| get_configs() (AFL.automation.instrument.I22SAXS.Driget_de   |  |
| method), 200   | method), 549   |
| get_configs() (AFL.automation.instrument.I22SAXS.I22\$\delta \text{XS}\delta \text{delta}  |  |
| method), 203   |  |
|  | method), 73, 694, 711  |
| get_configs() (AFL.automation.instrument.SeabreezeUVbyex_Dte   |  |
| method), 207   | method), 65, 419, 471  |
| get_configs() (AFL.automation.instrument.SeabreezeUVbyexSebe   |  |
| method), 217   | method), 509   |
| get_configs()(AFL.automation.instrument.SpecScreen_lgertedE  |  |
| method), 220   | (AFL.automation.APIServer.APIServer.APIServer  |
| <pre>get_configs() (AFL.automation.instrument.SpecScreen_Driver.S</pre>  | •  |
|  | river_object()   |
| $\verb"get_configs" () (AFL. automation. loading. Load Stopper Driver. Dr$   |  |
| method), 248   | method), 21, 162, 165  |
| <pre>get_configs() (AFL.automation.loading.LoadStopperDrigertLoad</pre>  |  |
| method), 251   | (AFL. automation. loading. Load Stopper Driver. Client   |
| <pre>get_configs() (AFL.automation.loading.OneSelectorBlowoutSan</pre>   |  |
|  | river_object()   |
| $\verb"get_configs"()" (AFL. automation. loading. One Selector Blowout Sandard Selector)" (AFL. automation. loading. One Selector) (AFL. automation. loading. One S$   |  |
| method), 273   | method), 418   |
| <pre>get_configs() (AFL.automation.loading.OneSelectorBlovgettSdx</pre>  | r <b>ipleCeBHyeoSe()</b> ctorBlowoutSampleCell   |

| (AFL.automation.prepare.OT2Client.Client   |                               | method), 578   |
|--|-------------------------------|--|
| method), 482   | get_int                       | eract_value()  |
| get_driver_object()  | J                             | (AFL.automation.prepare.SampleSeriesWidget.FloatText   |
| (AFL.automation.prepare.OT2Client.OT2Client  |                               | method), 589   |
| method), 485   | get_int                       | eract_value()  |
| get_driver_object()  | 5 –                           | (AFL.automation.prepare.SampleSeriesWidget.IntText     |
| (AFL.automation.prepare.SampleSeriesWidget.C   | lient                         | method), 604   |
| method), 585   |                               | eract_value()  |
| get_driver_object()  | 5 –                           | (AFL.automation.prepare.SampleSeriesWidget.Label       |
| (AFL.automation.sample.CastingServer.Client  |                               | method), 613   |
| method), 724   | get_int                       | eract_value()  |
| get_driver_object()  | 5 –                           | (AFL.automation.prepare.SampleSeriesWidget.Text        |
| (AFL.automation.sample.CastingServer.OT2Clie   | nt                            | method), 634   |
| method), 729   |                               | eract_value()  |
| get_feature_names_out()  | J                             | (AFL.automation.prepare.SweepBuilderWidget.Checkbox    |
| (AFL.automation.shared.DataLabelerWidget.Ora   | linalEnco                     |  |
| method), 746   |                               | eract_value()  |
| get_feature_names_out()  | 5 –                           | (AFL.automation.prepare.SweepBuilderWidget.Label       |
| (AFL.automation.shared.DiffractionLabeler.Ordi   | nalEncod                      |  |
| method), 767   |                               | eract_value()  |
| get_historical_values()  | J                             | (AFL.automation.prepare.SweepBuilderWidget.Text        |
| (AFL.automation.APIServer.Driver.PersistentCor   | ıfig                          | method), 699   |
| method), 171   |                               | _identity() (in module                                 |
| get_historical_values()  | 5 –5                          | AFL.automation.APIServer.APIServer), 92                |
|  | e <b>q#</b> Cto <i>i</i> k/wa | args() (AFL.automation.prepare.OT2Client.PipetteAction |
| method), 84, 775, 776  | 2 ,0                          | method), 487   |
| get_info()(AFL.automation.APIServer.APIServer.APISe  | ngvert_kwa                    |  |
| method), 18, 99, 157   | J                             | method), 387   |
| get_interact_value()   | get_lab                       | ware()(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_0 |
| (AFL.automation.prepare.DeckBuilderWidget.Ch   |                               |  |
| method), 411   |                               | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.Button       |
| (AFL.automation.prepare.DeckBuilderWidget.Dr   | opdown                        |  |
| method), 424   | _                             | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.Checkbox     |
| (AFL.automation.prepare.DeckBuilderWidget.La   | bel                           | static method), 411                                    |
| method), 440   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.Dropdown     |
| (AFL.automation.prepare.DeckBuilderWidget.Tex  | xt                            | static method), 424                                    |
| method), 459   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.HBox         |
| (AFL. automation. prepare. Prepare Widget. Checkber Minimum Checkber Min | ox                            | static method), 432                                    |
| method), 504   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.Label        |
| (AFL. automation. prepare. Prepare Widget. Dropdomation and the properties of the  | own                           | static method), 440                                    |
| method), 513   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL.automation.prepare.DeckBuilderWidget.Layout       |
| (AFL. automation. prepare. Prepare Widget. Label   |                               | static method), 451                                    |
| method), 529   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL. automation. prepare. Deck Builder Widget. Text   |
| (AFL. automation. prepare. Prepare Widget. Text  |                               | static method), 459                                    |
| method), 553   | get_man                       | ager_state()   |
| get_interact_value()   |                               | (AFL. automation. prepare. Deck Builder Widget. VBox   |
| (AFL automation prepare SampleSeriesWidget C.  | heckhox                       | static method) 466                                     |

| <pre>get_manager_state()      (AFL.automation.prepare.PrepareWidget.Button</pre> | <pre>get_manager_state()      (AFL.automation.prepare.SweepBuilderWidget.Checkbox</pre> |
|--|---|
| static method), 496  | static method), 662   |
| get_manager_state()  | get_manager_state()   |
| (AFL.automation.prepare.PrepareWidget.Checkl                                     |   |
| static method), 504  | static method), 670   |
| <pre>get_manager_state()</pre>   | get_manager_state()   |
| (AFL.automation.prepare.PrepareWidget.Dropdo                                     |   |
| static method), 513  | static method), 678   |
| <pre>get_manager_state()</pre>   | <pre>get_manager_state()</pre>  |
| (AFL.automation.prepare.PrepareWidget.HBox                                       | (AFL.automation.prepare.SweepBuilderWidget.Layout                                       |
| static method), 521  | static method), 689   |
| <pre>get_manager_state()</pre>   | <pre>get_manager_state()</pre>  |
| (AFL.automation.prepare.PrepareWidget.Label                                      | (AFL.automation.prepare.SweepBuilderWidget.Text   |
| static method), 529  | static method), 699   |
| <pre>get_manager_state()</pre>   | <pre>get_manager_state()</pre>  |
| (AFL.automation.prepare.PrepareWidget.Layout                                     |   |
| static method), 540  | static method), 706   |
| <pre>get_manager_state()</pre>   | get_metadata_routing()  |
| (AFL.automation.prepare.PrepareWidget.Text                                       | (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder                                 |
| static method), 553  | method), 746  |
| <pre>get_manager_state()</pre>   | get_metadata_routing()  |
| (AFL.automation.prepare.PrepareWidget.VBox                                       | (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder                                |
| static method), 560  | method), 767  |
| get_manager_state()  | get_name() (AFL.automation.APIServer.APIServer.FileHandler                              |
| (AFL.automation.prepare.SampleSeriesWidget.B                                     |   |
| static method), 570  | get_name() (AFL.automation.APIServer.APIServer.ServiceInfo                              |
| get_manager_state()  | method), 151  |
|  | hgekbaname() (AFL.automation.APIServer.APIServer.SMTPHandler                            |
| static method), 578  | method), 146  |
| <pre>get_manager_state()</pre>   | get_name() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo                      |
| (AFL.automation.prepare.SampleSeriesWidget.F.                                    |   |
| static method), 589  | get_name() (AFL.automation.shared.ServerDiscovery.ServiceInfo                           |
| get_manager_state()  | method), 801  |
| (AFL.automation.prepare.SampleSeriesWidget.H                                     |   |
| static method), 596  | (AFL.automation.shared.DatasetWidget.DatasetWidget_Model                                |
| <pre>get_manager_state()</pre>   | method), 79, 753, 758   |
|  | at <b>get</b> _object() (AFL.automation.APIServer.Client.Client                         |
| static method), 605  | method), 21, 162, 165   |
| <pre>get_manager_state()</pre>   | get_object() (AFL.automation.APIServer.Driver.Driver                                    |
| (AFL.automation.prepare.SampleSeriesWidget.La                                    |   |
| static method), 613  | get_object()(AFL.automation.APIServer.DummyDriver.Driver                                |
| <pre>get_manager_state()</pre>   | method), 175  |
|  | ayett_object() (AFL.automation.APIServer.DummyDriver.DummyDriver                        |
| static method), 623  | method), 178  |
| <pre>get_manager_state()</pre>   | <pre>get_object() (AFL.automation.APIServer.DummyOT2Driver.Driver</pre>                 |
| (AFL.automation.prepare.SampleSeriesWidget.Te                                    |   |
| static method), 634  | <pre>get_object() (AFL.automation.APIServer.DummyOT2Driver.DummyDri</pre>               |
| <pre>get_manager_state()</pre>   | method), 185  |
|  | Bget_object() (AFL.automation.instrument.DummySAS.Driver                                |
| static method), 641  | method), 195  |
| <pre>get_manager_state()</pre>   | get_object() (AFL.automation.instrument.DummySAS.DummySAS                               |
| (AFL.automation.prepare.SweepBuilderWidget.B                                     |   |
| static method), 654  | get_object()(AFL.automation.instrument.I22SAXS.Driver                                   |

| method), 201   | method), 585  |
|--|---|
| get_object()(AFL.automation.instrument.I22SAXS.I22SA <b>y&amp;</b> \$_obj  | ect() (AFL.automation.sample.CastingServer.CastingServer                            |
| method), 203   | method), 720  |
| get_object()(AFL.automation.instrument.SeabreezeUVVigeDrinkri  | ect() (AFL.automation.sample.CastingServer.Client                                   |
| method), 207   | method), 724  |
| get_object() (AFL.automation.instrument.SeabreezeUVVige.Seabrie  | ecal(YVAFL.automation.sample.CastingServer.Driver                                   |
| method), 217   | method), 726  |
| get_object()(AFL.automation.instrument.SpecScreen_Dgietr_Dbrig   | ect() (AFL.automation.sample.CastingServer.OT2Client                                |
| method), 221   | method), 729  |
| get_object()(AFL.automation.instrument.SpecScreen_Dgietr_Spec  |   |
| method), 223   | method), 733  |
| <pre>get_object() (AFL.automation.loading.LoadStopperDrivegetLiebt)</pre>  |   |
| method), 247   | method), 735  |
| get_object()(AFL.automation.loading.LoadStopperDrivegetDrivar  |   |
| method), 248   | method), 746  |
| get_object()(AFL.automation.loading.LoadStopperDrivgetoapts  |   |
| method), 251   | method), 767  |
| get_object() (AFL.automation.loading.OneSelectorBlow@werSapped   | **  |
| method), 269   | method), 77, 738, 748   |
| get_object() (AFL.automation.loading.OneSelectorBlowogerSapped   |   |
| method), 273   | method), 81, 761, 769   |
| get_object() (AFL.automation.loading.OneSelectorBlowogerSapppel  |   |
| method), 278   | <b>рсындаж</b> ыны адамын адамын кет. Биттуо 12 Бигет. Бит<br>method), 24, 184, 186 |
|  |   |
| get_object() (AFL.automation.loading.PneumaticPressugeStumptle   |   |
| method), 283   | method), 66, 477, 479   |
| get_object() (AFL.automation.loading.PneumaticPressugeStumple  |   |
| method), 287   | method), 18, 100, 158   |
| get_object() (AFL.automation.loading.PneumaticSampleGetl_Que   |   |
| method), 293   | method), 20, 161, 164   |
| get_object() (AFL.automation.loading.PneumaticSampleGetl_ffue  |   |
| method), 297   | method), 246  |
| get_object() (AFL.automation.loading.PushPullSelectorSymtplp(G   |   |
| method), 308   | method), 418  |
| get_object() (AFL.automation.loading.PushPullSelectorSymtplnG  | * *   |
| method), 312   | method), 481  |
| get_object() (AFL.automation.loading.RSoXSSolutionSagapteQube  |   |
| method), 319   | method), 485  |
| get_object()(AFL.automation.loading.RSoXSSolutionSagpteQube  |   |
| method), 323   | method), 584  |
| <pre>get_object() (AFL.automation.loading.TwoSelectorBlow@refSaquel</pre>  | · · · · · · · · · · · · · · · · · · ·   |
| method), 365   | method), 723  |
| get_object()(AFL.automation.loading.TwoSelectorBlow@etSaquet   | vie (I).[AstoBalutoondriovasatisphaplaGtillgServer.OT2Client                        |
| method), 370   | method), 729  |
| $\verb"get_object()" (AFL. automation. prepare. Deck Builder Widge \textit{Clique})$   | ue_iteration()  |
| method), 419   | (AFL.automation.APIServer.APIServer.APIServer                                       |
| <pre>get_object() (AFL.automation.prepare.Dummy_OT2_Driver.Driver</pre>  | emethod), 19, 100, 158  |
| method), 474 get_que   | ued_commands()  |
| $\verb"get_object()" (AFL. automation. prepare. Dummy\_OT2\_Driver. Dum$ | n( <u>A.FQT@ut</u> D <b>ni</b> ortion.APIServer.APIServer.APIServer                 |
| method), 478   | method), 18, 99, 158  |
| <pre>get_object() (AFL.automation.prepare.OT2Client.Clientget_que</pre>  | ued_commmands()   |
| method), 482   | (AFL.automation.APIServer.Client.Client   |
| <pre>get_object() (AFL.automation.prepare.OT2Client.OT2Client</pre>  | method), 20, 161, 164   |
|  | ued_commmands()   |
| $\verb"get_object()" (AFL. automation. prepare. Sample Series Widget. Clienter and the sample Serie$   | t (AFL.automation.loading.LoadStopperDriver.Client                                  |

| I D 046   | I D 202   |
|---|---|
| method), 246  | method), 203  |
| <pre>get_queued_commands()</pre>                                  | get_sample() (AFL.automation.instrument.SeabreezeUVVis.Driver ient method), 207   |
| method), 418  | get_sample() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUV  |
| get_queued_commmands()  | method), 217  |
| (AFL.automation.prepare.OT2Client.Client                          | get_sample() (AFL.automation.instrument.SpecScreen_Driver.Driver  |
| method), 482  | method), 221  |
| get_queued_commmands()  | get_sample() (AFL.automation.instrument.SpecScreen_Driver.SpecScree   |
| (AFL.automation.prepare.OT2Client.OT2Client                       | method), 223  |
| method), 485  | get_sample() (AFL.automation.loading.LoadStopperDriver.Driver   |
| <pre>get_queued_commmands()</pre>                                 | method), 248  |
| - · ·   | liget_sample() (AFL.automation.loading.LoadStopperDriver.LoadStopper.   |
| method), 584  | method), 251  |
| <pre>get_queued_commmands()</pre>                                 | <pre>get_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell.</pre>   |
| (AFL.automation.sample.CastingServer.Client                       | method), 269  |
| method), 723  | <pre>get_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell.</pre>   |
| <pre>get_queued_commmands()</pre>                                 | method), 273  |
| (AFL.automation.sample.CastingServer.OT2Clie                      | $nget\_sample()\ (AFL. automation. loading. One Selector Blowout Sample Cell. Supplies the sample of the sample of$   |
| method), 729  | method), 278  |
| <pre>get_quickbar() (AFL.automation.APIServer.APIServer.A</pre>   | Ag $f e$ Se $f s$ ample () (AFL. automation. loading. Pneumatic Pressure Sample Cell. D   |
| method), 18, 99, 158  | method), 283  |
| <pre>get_quickbar() (AFL.automation.APIServer.Client.Client</pre> | ${\tt nget\_sample()}\ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pressure$ |
| method), 20, 161, 164   | method), 287  |
| $get\_quickbar()$ (AFL.automation.loading.LoadStopperL            | Projectr_Shimpile() (AFL.automation.loading.PneumaticSampleCell.Driver  |
| method), 246  | method), 293  |
|   | Viglett. 63imple() (AFL.automation.loading.PneumaticSampleCell.Pneumatic  |
| method), 418  | method), 297  |
|   | $iegnet_sample()$ (AFL. automation. loading. Push Pull Selector Sample Cell. Dri  |
| method), 482  | method), 308  |
|   | **Q@hierstample() (AFL.automation.loading.PushPullSelectorSampleCell.Pus  |
| method), 485  | method), 312  |
|   | W <b>jdyets&amp;tiptle()</b> (AFL.automation.loading.RSoXSSolutionSampleCell.Drive  |
| method), 584  | method), 319  |
|   | Genusample() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX   |
| method), 723  | method), 323 <b>QF1Csample()</b> (AFL.automation.loading.TwoSelectorBlowoutSampleCell   |
| method), 729  | method), 365  |
|   | get_sample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell   |
| method), 22, 168, 172   | method), 370  |
|   | : Denves ample() (AFL automation.prepare.Dummy_OT2_Driver.Driver  |
| method), 175  | method), 474  |
|   | : Detus Dummy_OT2_Driver.Dummy_O  |
| method), 178  | method), 478  |
|   | rget_Dsample() (AFL.automation.sample.CastingServer.CastingServer   |
| method), 182  | method), 720  |
|   | riget_DsamplyDrjv(AFL.automation.sample.CastingServer.Driver  |
| method), 185  | method), 726  |
|   | Orgetr_sample() (AFL.automation.sample_env.TemperatureDeck.Driver   |
| method), 195  | method), 733  |
|   | DigentySample() (AFL.automation.sample_env.TemperatureDeck.Temperatu  |
| method), 197  | method), 735  |
| <pre>get_sample() (AFL.automation.instrument.I22SAXS.Driv</pre>   | eget_scattering()(AFL.automation.shared.DatasetWidget.DatasetWidge  |

get\_sample() (AFL.automation.instrument.I22SAXS.I22SAy&f\_send\_file\_max\_age()

method), 80, 753, 758

method), 201

```
(AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                             method), 459
                      method), 128
                                                                                                                                      get_state() (AFL.automation.prepare.DeckBuilderWidget.VBox
get_sensor_config()
                                                                                                                                                             method), 467
                      (AFL.automation.loading.PneumaticPressureSampleCelleRneumuhiFBranstameStimpleCellre.PrepareWidget.Button
                      method), 44, 288, 291
                                                                                                                                                             method), 496
get_sensor_config()
                                                                                                                                      get_state() (AFL.automation.prepare.PrepareWidget.Checkbox
                      (AFL.automation.loading.PneumaticSampleCell.PneumaticSauthpletCell)4
                      method), 45, 297, 301
                                                                                                                                       get_state() (AFL.automation.prepare.PrepareWidget.Dropdown
get_server_time() (AFL.automation.APIServer.APIServer.APIServer.time(), 514
                      method), 19, 101, 159
                                                                                                                                       get_state() (AFL.automation.prepare.PrepareWidget.HBox
get\_server\_time() (AFL.automation.APIServer.Client.Client
                                                                                                                                                             method), 522
                      method), 20, 161, 164
                                                                                                                                       get_state() (AFL.automation.prepare.PrepareWidget.Label
get_server_time() (AFL.automation.loading.LoadStopperDriver.Ghethod), 530
                                                                                                                                       get_state() (AFL.automation.prepare.PrepareWidget.Layout
                      method), 246
get_server_time() (AFL.automation.prepare.DeckBuilderWidget.Ghiethtod), 540
                      method), 418
                                                                                                                                       get_state() (AFL.automation.prepare.PrepareWidget.Text
get_server_time() (AFL.automation.prepare.OT2Client.Client
                                                                                                                                                            method), 553
                      method), 482
                                                                                                                                      get_state() (AFL.automation.prepare.PrepareWidget.VBox
\verb"get_server_time()" (AFL. automation. prepare. OT 2 Client. OT 2 Client method), 561
                      method), 485
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.Button
get_server_time()(AFL.automation.prepare.SampleSeriesWidget.Galithatd), 570
                      method), 584
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.Checkbox
get_server_time() (AFL.automation.sample.CastingServer.Client method), 578
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.FloatText
                      method), 723
get_server_time() (AFL.automation.sample.CastingServer.OT2Clianthod), 589
                      method), 729
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.HBox
get_service_info() (AFL.automation.APIServer.APIServer.Zerocompthod), 597
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.IntText
                      method), 155
get_service_info() (AFL.automation.APIServer.Client.ServerDisancetroyd), 605
                      method), 163
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.Label
get_service_info() (AFL.automation.shared.ServerDiscovery.ServerDiscovery
                      method), 85, 792, 808
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.Layout
get_service_info() (AFL.automation.shared.ServerDiscovery.Zerovathpd), 623
                      method), 805
                                                                                                                                      get_state() (AFL.automation.prepare.SampleSeriesWidget.Text
get_shake_latch_status()
                                                                                                                                                             method), 634
                      (AFL.automation.prepare.Dummy OT2 Driver.Dawnys to T2 (Driver.Dawnys to T2 (Driver.Dawn)s (Drive
                      method), 66, 477, 479
                                                                                                                                                             method), 642
get_shake_rpm() (AFL.automation.prepare.Dummy_OT2gDrivstDatin)vADE2udDmixenion.prepare.SweepBuilderWidget.Button
                      method), 66, 477, 479
                                                                                                                                                             method), 655
get_shaker_temp() (AFL.automation.prepare.Dummy_OffetDstart Dummy_OffetDstart Dummy_
                      method), 66, 477, 479
                                                                                                                                                             method), 663
get_state() (AFL.automation.prepare.DeckBuilderWidgegButtonate() (AFL.automation.prepare.SweepBuilderWidget.HBox
                      method), 404
                                                                                                                                                             method), 670
\verb|get_state()| (AFL. automation. prepare. Deck Builder Widget \textit{Ebeskbete}()| (AFL. automation. prepare. Sweep Builder Widget. Label Propagation of the propagation
                      method), 412
                                                                                                                                                             method), 679
get_state() (AFL.automation.prepare.DeckBuilderWidgetget_state() (AFL.automation.prepare.SweepBuilderWidget.Layout
                      method), 425
                                                                                                                                                             method), 689
get_state() (AFL.automation.prepare.DeckBuilderWidgetBB atate() (AFL.automation.prepare.SweepBuilderWidget.Text
                      method), 433
                                                                                                                                                             method), 699
get_state() (AFL.automation.prepare.DeckBuilderWidgetyEtate() (AFL.automation.prepare.SweepBuilderWidget.VBox
                                                                                                                                                             method), 707
                      method), 441
get_state()(AFL.automation.prepare.DeckBuilderWidgegetaysatock() (AFL.automation.prepare.Deck method),
                      method), 451
                                                                                                                                                             384
get_state() (AFL.automation.prepare.DeckBuilderWidgettextstock_objects()
```

```
(AFL.automation.prepare.PrepareWidget.StockBuilderWidgetSpec() (AFL.automation.prepare.DeckBuilderWidget.Layout
                 method), 548
                                                                                                                               method), 451
get_stock_objects()
                                                                                                             get_view_spec() (AFL.automation.prepare.DeckBuilderWidget.Text
                  (AFL.automation.prepare.StockBuilderWidget.StockBuilderWietbetd), 459
                 method), 71, 648, 649
                                                                                                             get_view_spec() (AFL.automation.prepare.DeckBuilderWidget.VBox
get_stock_values()(AFL.automation.prepare.PrepareWidget.StockRethilder;Wfdlget
                 method), 548
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.Button
get_stock_values() (AFL.automation.prepare.StockBuilderWidgetn\u00e4te\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u00df\u
                  method), 72, 648, 650
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.Checkbox
get_sweep_data() (AFL.automation.prepare.PrepareWidget.Sweep Builded) Widget
                 method), 549
                                                                                                             \texttt{get\_view\_spec()} (AFL.automation.prepare.PrepareWidget.Dropdown
get_sweep_data() (AFL.automation.prepare.SweepBuilderWidget.SweetpBb)ildeAWidget
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.HBox
                 method), 73, 694, 711
get_temp() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_Driver.Dummy_tollow_D
                 method), 66, 477, 479
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.Label
get_unit_type()
                                                                                           module
                                                                                                                               method), 530
                 AFL.automation.shared.units), 87, 812
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.Layout
get_unqueued_commands()
                                                                                                                               method), 540
                  (AFL.automation.APIServer.APIServer.APIServerget_view_spec() (AFL.automation.prepare.PrepareWidget.Text
                 method), 18, 99, 158
                                                                                                                               method), 553
get_unqueued_commmands()
                                                                                                             get_view_spec() (AFL.automation.prepare.PrepareWidget.VBox
                  (AFL.automation.APIServer.Client.Client
                                                                                                                               method), 561
                 method), 20, 161, 164
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Button
get_unqueued_commmands()
                                                                                                                               method), 570
                 (AFL.automation.loading.LoadStopperDriver.Clieget_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Checkbo
                 method), 246
                                                                                                                               method), 578
get_unqueued_commmands()
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.FloatTe.
                  (AFL.automation.prepare.DeckBuilderWidget.Client
                                                                                                                               method), 589
                 method), 418
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.HBox
get_unqueued_commmands()
                                                                                                                               method), 597
                 (AFL.automation.prepare.OT2Client.Client
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.IntText
                 method), 482
                                                                                                                               method), 605
get_unqueued_commmands()
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Label
                  (AFL.automation.prepare.OT2Client.OT2Client
                                                                                                                               method), 613
                 method), 485
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Layout
get_unqueued_commmands()
                                                                                                                               method), 623
                  (AFL.automation.prepare.SampleSeriesWidget.Cliqut_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Text
                 method), 584
                                                                                                                               method), 634
get_unqueued_commmands()
                                                                                                             get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.VBox
                 (AFL.automation.sample.CastingServer.Client
                                                                                                                               method), 642
                 method), 723
                                                                                                             get_view_spec() (AFL.automation.prepare.SweepBuilderWidget.Button
get_unqueued_commmands()
                                                                                                                               method), 655
                  (AFL.automation.sample.CastingServer.OT2Clienget_view_spec() (AFL.automation.prepare.SweepBuilderWidget.Checkbo
                 method), 729
                                                                                                                               method), 663
get_view_spec() (AFL.automation.prepare.DeckBuilderWidget.HBox
                  method), 404
                                                                                                                               method), 671
get_view_spec() (AFL.automation.prepare.DeckBuilderWidget.Label
                                                                                                                               method), 679
                 method), 412
get_view_spec() (AFL.automation.prepare.DeckBuilderWijdetenDewpsbpec() (AFL.automation.prepare.SweepBuilderWidget.Layout
                  method), 425
                                                                                                                               method), 689
get_view_spec() (AFL.automation.prepare.DeckBuilderWijdgenHBnxspec() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                               method), 699
                 method), 433
```

*method*), 441

get\_view\_spec() (AFL.automation.prepare.DeckBuilderWidget.VBox

method), 707

getOpenPorts()

(in

```
get_wells() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_KTT2utOnivation.loading.ChemyxSyringePump),
                              method), 66, 477, 479
                                                                                                                                                                                                                        35, 228, 233
getChannels() (AFL.automation.loading.MultiChannelRegetPhritain@tuenkiReites()
                              method), 40, 262, 263
                                                                                                                                                                                                                        (AFL.automation.loading.ChemyxSyringePump.ChemyxConnecti
getChannels() (AFL.automation.loading.SainSmartRelay.MultiChannelRelay,36, 230, 234
                              method), 330
                                                                                                                                                                                        getParameters() (AFL.automation.loading.ChemyxSyringePump.Chemy.
getChannels() (AFL.automation.loading.SainSmartRelay.SainSmartRelay.), 36, 230, 234
                                                                                                                                                                                         getParameters() (AFL. automation. loading. PushPullSelectorSampleCell.
                              method), 51, 331, 332
getDisplacedVolume()
                                                                                                                                                                                                                        method), 48, 311, 317
                              (AFL.automation.loading.ChemyxSyringePump.ChemyxCth)deAtoth.automation.loading.CetoniMultiPosValve.CetoniMultiPos
                              method), 36, 230, 234
                                                                                                                                                                                                                        method), 34, 227
getElapsedTime() (AFL.automation.loading.ChemyxSyringertPhonp.QheArFlx.Qutmactiiom.loading.CetoniMultiPosValve.FlowSelector
                              method), 36, 230, 234
                                                                                                                                                                                                                        method), 227
getExposure() (AFL.automation.instrument.SeabreezeUV\setSeabv@x(AFV\xintomation.loading.DoubleViciMultiposSelector.DoubleV
                              method), 30, 216, 218
                                                                                                                                                                                                                        method), 37, 238, 240
getExposureDelay() (AFL.automation.instrument.SeabregeatP&Fis(SeaHFLequL&MVitson.loading.DoubleViciMultiposSelector.FlowSel
                              method), 30, 216, 218
                                                                                                                                                                                                                        method), 238
getFilename() (AFL.automation.instrument.SeabreezeUVWexBoodverteWintomation.loading.DoubleViciMultiposSelector.ViciMult
                              method), 30, 216, 218
                                                                                                                                                                                                                        method), 239
getFilepath() (AFL.automation.instrument.SeabreezeUVyetBeartneytelFEXintomation.loading.FlowSelector.FlowSelector
                              method), 30, 216, 218
                                                                                                                                                                                                                        method), 38, 243
getLevel() (AFL.automation.loading.ChemyxSyringePumg.ClPcortxSyringePumpmation.loading.ViciMultiposSelector.FlowSelector
                              method), 35, 231, 233
                                                                                                                                                                                                                        method), 378
getName() (AFL.automation.APIServer.APIServer.QueueDgenDoort() (AFL.automation.loading.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMultiposSelector.ViciMu
                              method), 143
                                                                                                                                                                                                                        method), 61, 379, 380
getName() (AFL.automation.APIServer.QueueDaemon.QuegetPump&ntatus() (AFL.automation.loading.ChemyxSyringePump.Chemy.
                              method), 191
                                                                                                                                                                                                                        method), 36, 230, 234
getName() (AFL.automation.EpicsADLiveProcess.ReduceDgetRart-ReduAHDacemomation.loading.ChemyxSyringePump.ChemyxSyringe
                                                                                                                                                                                                                        method), 35, 231, 233
                              method), 825
getName() (AFL.automation.loading.LoadStopperDriver.SagetReling(Arladutomation.loading.ChemyxSyringePump.SyringePump
                              method), 254
                                                                                                                                                                                                                        method), 232
getName() (AFL.automation.loading.LoadStopperDriver.SigetRatleBylAFL.automation.loading.DummyPump.DummyPump
                              method), 257
                                                                                                                                                                                                                        method), 38, 241, 243
getName() (AFL.automation.loading.LoadStopperDriver.StgetRatleBy2AFL.automation.loading.DummyPump.SyringePump
                              method), 260
                                                                                                                                                                                                                        method), 242
getName() (AFL.automation.loading.Sensor.DummySensorgetRate() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePump.
                              method), 335
                                                                                                                                                                                                                       method), 41, 265, 267
getName() (AFL.automation.loading.Sensor.DummySensorgetRate() (AFL.automation.loading.NE1kSyringePump.SyringePump
                              method), 338
                                                                                                                                                                                                                        method), 266
getName() (AFL.automation.loading.SensorCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThr
                                                                                                                                                                                                                       method), 46, 303, 305
                              method), 343
getName() (AFL.automation.loading.SensorCallbackThreagsSiBarteXfreAlEddatBomation.loading.PressureControllerAsPump.SyringeF
                                                                                                                                                                                                                        method), 305
                              method), 346
\texttt{getName()} \ (AFL. automation. loading. Sensor Callback Threa \textbf{getName()} \ (\textbf{\textit{RFL}}. automation. loading. Syringe Pump. 
                              method), 349
                                                                                                                                                                                                                        method), 58, 361
getName() (AFL.automation.loading.SensorCallbackThreads\Despon(SBQ) (AFL.automation.loading.ChemyxSyringePump.ChemyxC
                              method), 352
                                                                                                                                                                                                                        method), 36, 230, 233
getName() (AFL.automation.loading.SensorPollingThread.GensarVestingThescath()
                              method), 357
                                                                                                                                                                                                                        (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is\\
getName() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                        method), 30, 216, 218
                                                                                                                                                                                         \verb"getServerState"() (AFL. automation. loading. Sensor Callback Thread. Loading. Sensor Callback T
                              method), 790
getName() (AFL.automation.shared.ServerDiscovery.ServiceBrowsemethod), 56, 341, 355
                              method), 795
                                                                                                                                                                                         getStatus() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxS
```

872 Index

method), 35, 232, 233

module

```
getStatus() (AFL.automation.loading.DummyPump.DummyPump attribute), 450
        method), 38, 241, 243
                                                      grid\_column (AFL. automation. prepare. Prepare Widget. Layout
getStatus() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePump), 539
        method), 41, 265, 267
                                                      \verb"grid_column" (AFL. automation. prepare. Sample Series Widget. Layout
getStatus() (AFL.automation.loading.PressureControllerAsPump.RuesibutaSpm2fbllerAsPump
        method), 47, 304, 305
                                                      grid_column(AFL.automation.prepare.SweepBuilderWidget.Layout
getSubject() (AFL.automation.APIServer.APIServer.SMTPHandlerattribute), 688
                                                      grid_gap (AFL.automation.prepare.DeckBuilderWidget.Layout
        method), 146
getValueFromParams()
                                                               attribute), 449
        (AFL.automation.loading.ChemyxSyringePump.Charindxsgraph(getPlumptomation.prepare.PrepareWidget.Layout
        method), 35, 232, 233
                                                               attribute), 538
glob() (AFL.automation.instrument.SeabreezeUVVis.Path grid_gap(AFL.automation.prepare.SampleSeriesWidget.Layout
        method), 211
                                                               attribute), 621
got_first_request (AFL.automation.APIServer.APIServer.Edustary (AFL.automation.prepare.SweepBuilderWidget.Layout
                                                                attribute), 687
        property), 115
goto_callback() (AFL.automation.shared.DataLabelerWighted Down LAbdlanWidgetion.prepare.DeckBuilderWidget.Layout
                                                               attribute), 450
         method), 77, 740, 747
goto_callback() (AFL.automation.shared.DatasetWidgetgPattusetWilderElL.automation.prepare.PrepareWidget.Layout
         method), 79, 751, 757
                                                                attribute), 539
goto_callback() (AFL.automation.shared.DiffractionLabgkinDiffractionLabakindination.prepare.SampleSeriesWidget.Layout
        method), 81, 760, 768
                                                                attribute), 622
grid_area (AFL.automation.prepare.DeckBuilderWidget.Layout_row (AFL.automation.prepare.SweepBuilderWidget.Layout
         attribute), 450
                                                                attribute), 688
grid_area(AFL.automation.prepare.PrepareWidget.Layougrid_template_areas
                                                               (AFL.automation.prepare.DeckBuilderWidget.Layout
         attribute), 539
{\tt grid\_area}\,(AFL. automation. prepare. Sample Series Widget. Layout
                                                               attribute), 450
         attribute), 622
                                                      grid_template_areas
{\tt grid\_area} \, (AFL. automation. prepare. Sweep Builder Widget. Layout
                                                               (AFL.automation.prepare.PrepareWidget.Layout
        attribute), 688
                                                               attribute), 539
grid_auto_columns(AFL.automation.prepare.DeckBuildentVidlgtehpplatte_areas
         attribute), 449
                                                                (AFL.automation.prepare.SampleSeriesWidget.Layout
grid_auto_columns (AFL.automation.prepare.PrepareWidget.Layouttribute), 622
         attribute), 538
                                                      grid_template_areas
grid_auto_columns (AFL.automation.prepare.SampleSeriesWidget.Layout
         attribute), 621
                                                                attribute), 688
grid_auto_columns (AFL.automation.prepare.SweepBuildparWidgempdapane_columns
         attribute), 687
                                                               (AFL.automation.prepare.DeckBuilderWidget.Layout
grid_auto_flow(AFL.automation.prepare.DeckBuilderWidget.Layoattribute), 449
         attribute), 449
                                                      grid_template_columns
grid_auto_flow (AFL.automation.prepare.PrepareWidget.Layout (AFL.automation.prepare.PrepareWidget.Layout
        attribute), 538
                                                               attribute), 538
grid_auto_flow(AFL.automation.prepare.SampleSeriesWgdiget_Itaeynputate_columns
                                                               (AFL.automation.prepare.SampleSeriesWidget.Layout
         attribute), 621
grid_auto_flow(AFL.automation.prepare.SweepBuilderWidget.Layoutribute), 621
                                                      grid_template_columns
         attribute), 687
grid_auto_rows (AFL.automation.prepare.DeckBuilderWidget.Layout_AFL.automation.prepare.SweepBuilderWidget.Layout_
         attribute), 449
                                                               attribute), 687
grid_auto_rows (AFL.automation.prepare.PrepareWidgetdxixdytemplate_rows (AFL.automation.prepare.DeckBuilderWidget.Layo
         attribute), 538
                                                               attribute), 449
grid_auto_rows (AFL.automation.prepare.SampleSeriesWgdget_Itaeynputate_rows (AFL.automation.prepare.PrepareWidget.Layout
```

grid\_auto\_rows (AFL.automation.prepare.SweepBuilderWgriget\_Lampltate\_rows (AFL.automation.prepare.SampleSeriesWidget.Layo

grid\_column (AFL.automation.prepare.DeckBuilderWidgegtbinkontemplate\_rows (AFL.automation.prepare.SweepBuilderWidget.Lay

attribute), 621

attribute), 687

attribute), 538

attribute), 621

|                    | attribute), 687  |         | (AFL.automation.prepare.PrepareWidget.Button         |
|--------------------|--|---------|--|
| <pre>group()</pre> | (AFL. automation. instrument. Seabreeze UVV is. Pathologies and the property of the property | h       | static method), 496                                  |
|                    | <i>method</i> ), 211   | handle_ | comm_opened()  |
|                    |  |         | (AFL.automation.prepare.PrepareWidget.Checkbox       |
| Н                  |  |         | static method), 504                                  |
| halt()(            | AFL.automation.APIServer.APIServer.APIServer   | handle_ | comm_opened()  |
|                    | method), 19, 100, 159  |         | (AFL. automation. prepare. Prepare Widget. Drop down |
| halt()             | (AFL.automation.APIServer.Client.Client  |         | static method), 514                                  |
|                    | method), 20, 161, 165  | handle_ | comm_opened()  |
| halt()(            | AFL.automation.EpicsADLiveProcess.Client.Clien   | ıt .    | (AFL.automation.prepare.PrepareWidget.HBox           |
|                    | method), 27, 822, 823  |         | static method), 522                                  |
| halt()(            | AFL. automation. loading. Load Stopper Driver. Clientum St. Clientum | handle_ | comm_opened()  |
|                    | method), 246   |         | (AFL.automation.prepare.PrepareWidget.Label          |
| halt()(            | AFL. automation. prepare. Deck Builder Widget. Clier   | ıt      | static method), 530                                  |
|                    | method), 418   | handle_ | comm_opened()  |
| halt()             | (AFL.automation.prepare.OT2Client.Client   |         | (AFL.automation.prepare.PrepareWidget.Layout         |
|                    | method), 482   |         | static method), 540                                  |
| halt()             | (AFL.automation.prepare.OT2Client.OT2Client  | handle_ | comm_opened()  |
|                    | method), 485   |         | (AFL.automation.prepare.PrepareWidget.Text           |
| halt()(            | AFL. automation. prepare. Sample Series Widget. Clie   | ent     | static method), 553                                  |
|                    | method), 585   | handle_ | comm_opened()  |
| halt()             | (AFL. automation. sample. Casting Server. Client   |         | (AFL.automation.prepare.PrepareWidget.VBox           |
|                    | method), 724   |         | static method), 561                                  |
| halt()(            | AFL.automation.sample.CastingServer.OT2Client  | handle_ | comm_opened()  |
|                    | method), 729   |         | (AFL.automation.prepare.SampleSeriesWidget.Button    |
| handle(            | ) (AFL.automation.APIServer.APIServer.FileHand   | ller    | static method), 570                                  |
|                    | method), 104   | handle_ | comm_opened()  |
| handle(            | ) (AFL. automation. APIS erver. APIS erver. SMTPH $lpha$   | ındler  | (AFL.automation.prepare.SampleSeriesWidget.Checkbox  |
|                    | method), 147   |         | static method), 578                                  |
| handle_            | comm_opened()  | handle_ | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Builder Widg | tton    | (AFL.automation.prepare.SampleSeriesWidget.FloatText |
|                    | static method), 404  |         | static method), 589                                  |
| handle_            |  |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Change the properties of the prope | eckbox  | (AFL.automation.prepare.SampleSeriesWidget.HBox      |
|                    | static method), 412  |         | static method), 597                                  |
| handle_            | comm_operied()   |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Dream and the prepare of the pr | opdown  | (AFL.automation.prepare.SampleSeriesWidget.IntText   |
|                    | static method), 425  |         | static method), 605                                  |
| handle_            | comm_operied()   |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. HB   | ox      | (AFL.automation.prepare.SampleSeriesWidget.Label     |
|                    | static method), 433  |         | static method), 613                                  |
| handle_            |  |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Laboration and the prepare of t | bel     | (AFL.automation.prepare.SampleSeriesWidget.Layout    |
|                    | static method), 441  |         | static method), 623                                  |
| handle_            |  |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Lay and the properties of the prop | vout    | (AFL.automation.prepare.SampleSeriesWidget.Text      |
|                    | static method), 451  |         | static method), 634                                  |
| handle_            | comm_operied()   |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. Texture of the prepare of the p | rt .    | (AFL.automation.prepare.SampleSeriesWidget.VBox      |
|                    | static method), 459  |         | static method), 642                                  |
| handle_            |  |         | comm_opened()  |
|                    | (AFL. automation. prepare. Deck Builder Widget. VB-1000000000000000000000000000000000000   | ox      | (AFL.automation.prepare.SweepBuilderWidget.Button    |
|                    | static method), 467  | ,       | static method), 655                                  |
| handle             | comm opened()  | nandle_ | comm_opened()  |

| (AFL. automation. prepare. Sweep Builder Widget. Check box        |   |
|---|---|
| static method), 663   | class method), 530  |
| handle_comm_opened() handle                                       | e_control_comm_opened()   |
| (AFL. automation. prepare. Sweep Builder Widget. HBox             | (AFL. automation. prepare. Prepare Widget. Layout                   |
| static method), 671   | class method), 540  |
| handle_comm_opened() handle                                       | e_control_comm_opened()   |
| (AFL. automation. prepare. Sweep Builder Widget. Label            | (AFL. automation. prepare. Prepare Widget. Text                     |
| static method), 679   | class method), 553  |
| handle_comm_opened() handle                                       | e_control_comm_opened()   |
| (AFL. automation. prepare. Sweep Builder Widget. Layout           | (AFL.automation.prepare.PrepareWidget.VBox                          |
| static method), 689   | class method), 561  |
| handle_comm_opened() handle                                       | e_control_comm_opened()   |
| (AFL. automation. prepare. Sweep Builder Widget. Text             | (AFL.automation.prepare.SampleSeriesWidget.Button                   |
| static method), 699   | class method), 570  |
| handle_comm_opened() handle                                       | e_control_comm_opened()   |
| (AFL.automation.prepare.SweepBuilderWidget.VBox                   | (AFL.automation.prepare.SampleSeriesWidget.Checkbox                 |
| static method), 707   | class method), 578  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Button                  | (AFL.automation.prepare.SampleSeriesWidget.FloatText                |
| class method), 404  | class method), 589  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Checkbox                | (AFL.automation.prepare.SampleSeriesWidget.HBox                     |
| class method), 412  | class method), 597  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Dropdown                | - · · · · · · · · · · · · · · · · · · ·                             |
| class method), 425  | class method), 605  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.HBox                    | (AFL.automation.prepare.SampleSeriesWidget.Label                    |
| class method), 433  | class method), 613  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Label                   | (AFL.automation.prepare.SampleSeriesWidget.Layout                   |
| class method), 441  | class method), 623  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Layout                  | (AFL.automation.prepare.SampleSeriesWidget.Text                     |
| class method), 451  | class method), 634  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.Text                    | (AFL.automation.prepare.SampleSeriesWidget.VBox                     |
| class method), 459  | class method), 642  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.DeckBuilderWidget.VBox                    | (AFL.automation.prepare.SweepBuilderWidget.Button                   |
| class method), 467  | class method), 655  |
|   | e_control_comm_opened()   |
|   | (AFL.automation.prepare.SweepBuilderWidget.Checkbox                 |
| (AFL.automation.prepare.PrepareWidget.Button class method), 496   | class method), 663  |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.PrepareWidget.Checkbox                    | (AFL.automation.prepare.SweepBuilderWidget.HBox                     |
| class method), 504  | class method), 671  |
|   | e_control_comm_opened()   |
|   | - · · · · · · · · · · · · · · · · · · ·                             |
| (AFL.automation.prepare.PrepareWidget.Dropdown class method), 514 | (AFL.automation.prepare.SweepBuilderWidget.Label class method), 679 |
|   | e_control_comm_opened()   |
| (AFL.automation.prepare.PrepareWidget.HBox                        | (AFL.automation.prepare.SweepBuilderWidget.Layout                   |
| class method), 522  | class method), 689  |
|   |   |
| handle_control_comm_opened() handle                               | e_control_comm_opened()   |

| (AFL. automation. prepare. Sweep Builder Widget. T   | <i>e</i> lmas_tra              | it()(AFL.autor                  | mation.prepare.P    | repareWidget.VI                       | Box           |
|--|--------------------------------|---------------------------------|---------------------|---------------------------------------|---------------|
| class method), 699   |                                | <i>method</i> ), 561            |                     |                                       |               |
| handle_control_comm_opened()   | has_tra                        | it()(AFL.autor                  | mation.prepare.Sc   | ampleSeriesWid <sub>e</sub>           | get.Button    |
| (AFL. automation. prepare. Sweep Builder Widget. Value of the property of th     | Box                            | method), 570                    |                     |                                       |               |
| class method), 707   |                                |                                 | nation.prepare.Sc   | ample Series Wid                      | get.Checkbox  |
| ${\tt handle\_exception()}\ (AFL. automation. APIServer. APISe$  |                                |                                 |                     |                                       |               |
| method), 123   | has_tra                        |                                 | nation.prepare.Se   | ample Series Wid,                     | get.FloatText |
| handle_http_exception()  |                                | method), 589                    |                     |                                       |               |
| (AFL.automation.APIServer.APIServer.Flask method), 122   | has_tra                        | it() (AFL.autor<br>method), 597 | nation.prepare.So   | ampleSeriesWid <sub>e</sub>           | get.HBox      |
| handle_url_build_error()   | has_tra                        | it()(AFL.autor                  | mation.prepare.Se   | ampleSeriesWid <sub>e</sub>           | get.IntText   |
| (AFL. automation. API Server. API Server. Flask  |                                | method), 605                    |                     |                                       |               |
| method), 132   | has_tra                        | it()(AFL.autor                  | nation.prepare.Sc   | ample Series Wid                      | get.Label     |
| handle_user_exception()  |                                | <i>method</i> ), 613            |                     |                                       |               |
| (AFL.automation.APIServer.APIServer.Flask method), 122   | has_tra                        | it() (AFL.autor<br>method), 623 | nation.prepare.So   | ampleSeriesWid <sub>e</sub>           | get.Layout    |
| handleError()(AFL.automation.APIServer.APIServer.F   | il <b>hÆs<u>n</u>tdka</b>      | it()(AFL.autor                  | nation.prepare.Se   | ampleSeriesWid,                       | get.Text      |
| method), 104   |                                | <i>method</i> ), 634            |                     |                                       |               |
| ${\tt handleError()}\ (AFL. automation. APIS erver. APIS erver. Shape of the automation of the automatic of the automation of the automatio$ | MHASH turd                     | liet () (AFL.autor              | nation.prepare.Sc   | ample Series Wid                      | get.VBox      |
| method), 147   |                                | <i>method</i> ), 642            |                     |                                       |               |
| ${\tt hardlink\_to()}\ (AFL. automation. instrument. Seabreeze U$  | VN/ais <i>Paut</i> ha          |                                 | nation.prepare.S    | weepBuilderWid                        | get.Button    |
| method), 212   |                                | method), 655                    |                     |                                       |               |
| ${\tt has\_static\_folder} (AFL. automation. API Server. API Ser$  | v <b>lava.E.[</b> atsika       |                                 | nation.prepare.S    | weepBuilderWid                        | get.Checkbox  |
| property), 128   |                                | <i>method</i> ), 663            |                     |                                       |               |
| ${\tt has\_trait()}$ (AFL.automation.prepare.DeckBuilderWidg   | e <b>h<i>Bstt</i>on</b> a      |                                 | mation.prepare.S    | weepBuilderWid                        | get.HBox      |
| method), 404   |                                | <i>method</i> ), 671            |                     |                                       |               |
| has_trait()(AFL.automation.prepare.DeckBuilderWidg   | e <b>hé</b> sethda             |                                 | nation.prepare.S    | weepBuilderWid                        | get.Label     |
| method), 412   |                                | method), 679                    |                     | D 41.1 YYY                            |               |
| has_trait() (AFL.automation.prepare.DeckBuilderWidg  | e <b>nes</b> optata            |                                 | mation.prepare.S    | weepBuilderWid                        | get.Layout    |
| method), 425   | 1.770 .                        | method), 689                    |                     | D 111 TW:                             |               |
| has_trait() (AFL.automation.prepare.DeckBuilderWidg<br>method), 433  |                                | method), 699                    | • •                 | •                                     |               |
| <pre>has_trait() (AFL.automation.prepare.DeckBuilderWidg</pre>   | e <b>hās<u>b</u>at</b> ra      | it() (AFL.autor<br>method), 707 | nation.prepare.S    | weepBuilderWid                        | get.VBox      |
| has_trait()(AFL.automation.prepare.DeckBuilderWidg   | e <b>hAs<u>y</u>ww</b> i       | ts()                            | (in                 | module                                |               |
| method), 451   |                                | AFL.automation                  | n.prepare.factory   | ), 712                                |               |
| ${\tt has\_trait()}$ (AFL.automation.prepare.DeckBuilderWidg   | <i>ehTes<u>xt</u>u</i> ni      | ts() (in module                 | e AFL.automation    | ı.shared.units),                      |               |
| method), 459   |                                | 87, 812                         |                     |                                       |               |
| has_trait() (AFL.automation.prepare.DeckBuilderWidg  | <i>eh₩<u>B</u>o</i> umi        |                                 | (in                 | module                                |               |
| method), 467   |                                |                                 | n.shared.utilities) | *                                     |               |
| has_trait() (AFL.automation.prepare.PrepareWidget.Bi   | uthBox (cla                    |                                 | ation.prepare.De    | ckBuilderWidge                        | <i>t</i> ),   |
| method), 496   |                                | 429                             |                     |                                       |               |
| has_trait()(AFL.automation.prepare.PrepareWidget.Cl  | n <b>eHBoon</b> x(clo          |                                 | nation.prepare.P    | repareWidget),                        |               |
| method), 504   | ····· / 1                      | 518                             |                     | 1.6                                   |               |
| has_trait()(AFL.automation.prepare.PrepareWidget.Di  | co <b>instan</b> tation of the |                                 | atıon.prepare.Sai   | npleSeriesWidge                       | et),          |
| method), 514   | Dun ( 1                        | 593                             |                     | D :11 HV:1                            |               |
| has_trait() (AFL.automation.prepare.PrepareWidget.H.<br>method), 522   | RHROX (Cla                     | iss in AFL.autom<br>667         | ation.prepare.Sw    | еерВинаегWнад                         | et),          |
| has_trait() (AFL.automation.prepare.PrepareWidget.La   | и <b>НШ</b> 20Fac              |                                 | dule AFL.autom      | ation.prepare)                        |               |
| method), 530   |                                | 381                             |                     | · · · · · · · · · · · · · · · · · · · |               |
| has_trait() (AFL.automation.prepare.PrepareWidget.La   | и <b>Шии</b> OFac              |                                 | (in                 | module                                |               |
| method), 540   | •                              |                                 | n.prepare.factory   |                                       |               |
| has_trait() (AFL.automation.prepare.PrepareWidget.Te   | xt                             | 716                             |                     |                                       |               |
| method), 553   |                                | AFL.automation                  | .prepare.DeckBu     | ilderWidget.Lav                       | out           |

| attribute), 448   | method), 634  |
|---|---|
| height ( <i>AFL.automation.prepare.PrepareWidget.Layout</i> hold_sy attribute), 537       |   |
| height (AFL.automation.prepare.SampleSeriesWidget.Laydhold_sy attribute), 620             |   |
| height (AFL.automation.prepare.SweepBuilderWidget.Laybrald_sy attribute), 686             |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgehBlutosy<br>method), 404              |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgeh6hdcksyncethod), 412                 |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgehDrdpdy<br>method), 425               |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgehb\bbssymethod), 433                  |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgehbattelsy method), 441                |   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidgehblyfortr<br>method), 451             | ait_notifications()   |
| $\verb+hold_sync()+ (AFL. automation. prepare. Deck Builder Widget. Text$                 | (AFL.automation.prepare.DeckBuilderWidget.Button method), 404             |
|   | ait_notifications()   |
| hold_sync() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 467                   | (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 412           |
| hold_sync() (AFL.automation.prepare.PrepareWidget.Buthold_tr<br>method), 496              | ait_notifications()<br>(AFL.automation.prepare.DeckBuilderWidget.Dropdown |
| hold_sync() (AFL.automation.prepare.PrepareWidget.Checkbox<br>method), 504 hold_tr        | <pre>method), 425 ait_notifications()</pre>                               |
| hold_sync() (AFL.automation.prepare.PrepareWidget.Dropdown method), 514                   | (AFL.automation.prepare.DeckBuilderWidget.HBox method), 433               |
| $\verb hold_sync()  (AFL. automation. prepare. PrepareWidget. HB \verb hold_tr ) $        |   |
| method), 522  | (AFL.automation.prepare.DeckBuilderWidget.Label                           |
| hold_sync() (AFL.automation.prepare.PrepareWidget.Label                                   | method), 441  |
| method), 530 hold_tr<br>hold_sync() (AFL.automation.prepare.PrepareWidget.Layout          | ait_notifications() (AFL.automation.prepare.DeckBuilderWidget.Layout      |
| method), 540  | method), 451  |
| hold_sync() (AFL.automation.prepare.PrepareWidget.Texhold_tr                              |   |
| method), 553  | (AFL.automation.prepare.DeckBuilderWidget.Text                            |
| hold_sync() (AFL.automation.prepare.PrepareWidget.VBox                                    | method), 459  |
| method), 561 hold_tr  | ait_notifications()   |
| ${\tt hold\_sync()}~(AFL. automation. prepare. Sample Series Widget. Button~method), 570$ | (AFL.automation.prepare.DeckBuilderWidget.VBox method), 467               |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidgetoChleckst<br>method), 578           | wit_notifications() (AFL.automation.prepare.PrepareWidget.Button          |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidget.FloatTemperator), 589 hold_tr      | emtethod), 496 ait_notifications()  |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 597                  | (AFL.automation.prepare.PrepareWidget.Checkbox method), 504               |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidgerolmdText                            |   |
| method), 605  | (AFL.automation.prepare.PrepareWidget.Dropdown                            |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidget.Label                              | method), 514 ait_notifications()  |
| <pre>method), 613</pre>   |   |
| method), 623  | method), 522  |
| hold_sync() (AFL.automation.prepare.SampleSeriesWidghtoTkdttr                             |   |

| (AFL.automation.prepare.PrepareWidget.Label method), 530   |                  | (AFL.automation.prepare.SweepBuilderWidget.Text method), 699              |
|--|------------------|---|
|  | hold_tr          | rait_notifications()  |
| (AFL.automation.prepare.PrepareWidget.Layout   |                  | (AFL.automation.prepare.SweepBuilderWidget.VBox                           |
| method), 540   |                  | method), 707  |
|  | home()(          | (AFL.automation.EpicsADLiveProcess.Client.Client                          |
| (AFL.automation.prepare.PrepareWidget.Text   |                  | method), 27, 822, 823   |
|  | home()(          | (AFL.automation.instrument.SeabreezeUVVis.Path                            |
| hold_trait_notifications()   | 1101110 () (     | class method), 211  |
|  | home()(          | (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driv                   |
| method), 561   |                  | method), 66, 477, 479   |
|  |                  | (AFL.automation.prepare.OT2Client.OT2Client                               |
| (AFL.automation.prepare.SampleSeriesWidget.But   |                  | method), 67, 485, 487   |
|  | nome()(          | (AFL.automation.sample.CastingServer.OT2Client                            |
| hold_trait_notifications()   |                  | method), 729  |
|  | <i>aokhax</i> tt | 1 (AFL.automation.APIServer.APIServer.ServiceInfo                         |
| method), 578   | _                | attribute), 149   |
| hold_trait_notifications()   |                  | 1 (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo attribute), 781 |
| method), 589   | host_tt          | :1 (AFL.automation.shared.ServerDiscovery.ServiceInfo                     |
| hold_trait_notifications()   |                  | attribute), 799   |
|  | how_man          | y() (AFL.automation.APIServer.DummyDriver.DummyDriver                     |
| method), 597   |                  | method), 23, 178, 179   |
|  | how_man          | y() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver                  |
| (AFL.automation.prepare.SampleSeriesWidget.Int   |                  | method), 24, 184, 186   |
| method), 605   |                  | ,, , ,  |
| hold_trait_notifications()   |                  |   |
| (AFL. automation. prepare. Sample Series Widget. Laboration and the property of the property | hebsays          | S (class in AFL automation instrument 122SAXS)                            |
| method), 613   | LLLJIIN          | 29, 202, 204  |
|  | icon(Al          | FL.automation.prepare.DeckBuilderWidget.Button                            |
| (AFL.automation.prepare.SampleSeriesWidget.Lay   | vout             | attribute), 402   |
| I D 600  |                  | FL.automation.prepare.PrepareWidget.Button at-                            |
| hold_trait_notifications()   | I COII (AI       | tribute), 494   |
|  | stoon (Al        | FL.automation.prepare.SampleSeriesWidget.Button                           |
| method), 634   | #Con (AI         |   |
|  | i aan (          | attribute), 568   |
| (AFL.automation.prepare.SampleSeriesWidget.VB  | icon (Ar         | FL.automation.prepare.SweepBuilderWidget.Button                           |
|  |                  | attribute), 653   |
| hold_trait_notifications()   | iaent (A         | AFL.automation.APIServer.APIServer.QueueDaemon                            |
|  | ##### . / A      | property), 143  |
| method), 655   | iwent (A         | AFL.automation.APIServer.QueueDaemon.QueueDaemon                          |
|  |                  | property), 191  |
|  |                  | AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaem                 |
| (AFL.automation.prepare.SweepBuilderWidget.Ch<br>method), 663  |                  | property), 825  |
| hold_trait_notifications()   | ident (A         | AFL.automation.loading.LoadStopperDriver.SensorPollingThread              |
|  |                  | property), 254  |
| (AFL.automation.prepare.SweepButtaerwaget.His  | ident (A         | AFL.automation.loading.LoadStopperDriver.StopLoadCBv1                     |
| method), 671   |                  | property), 257  |
|  |                  | NFL.automation.loading.LoadStopperDriver.StopLoadCBv2                     |
| (AFL.automation.prepare.SweepBuilderWidget.Lau   |                  | property), 260  |
| method), 679   | ident (          | AFL.automation.loading.Sensor.DummySensor1                                |
| hold_trait_notifications()   |                  | property), 335  |
| (AFL.automation.prepare.SweepBuilderWidget.Lag   | ¥θ⊌ht (          | AFL.automation.loading.Sensor.DummySensor2                                |
| method), 689   |                  | property), 338  |
| hold_trait_notifications()   |                  |   |

```
ident (AFL.automation.loading.SensorCallbackThread.SeninitalbackTflgendnifest()
                                                                                                                                                                                                                                                                           (AFL.automation.sample.CastingServer.CastingServer
                                     property), 343
ident (AFL.automation.loading.SensorCallbackThread.SimpleThreshmkd&\( \mathbb{R} \)), 89, 720, 730
                                                                                                                                                                                                                                    init_checkboxes() (AFL.automation.shared.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Datase
                                     property), 346
ident (AFL.automation.loading.SensorCallbackThread.StopLoadCBvNethod), 80, 754, 758
                                                                                                                                                                                                                                    init_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget
                                     property), 349
ident (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2ethod), 80, 754, 758
                                                                                                                                                                                                                                     init_inputs() (AFL.automation.shared.DatasetWidget.DatasetWidget_V
                                     property), 352
ident (AFL.automation.loading.SensorPollingThread.SensorPollingThmethdd), 80, 754, 758
                                                                                                                                                                                                                                     init_logging() (AFL.automation.APIServer.APIServer.APIServer
                                     property), 357
ident(AFL. automation. shared. Server Discovery. Run Thread
                                                                                                                                                                                                                                                                           method), 18, 100, 158
                                                                                                                                                                                                                                      init_models() (AFL.automation.shared.DataLabelerWidget.DataLabeler
                                     property), 790
{\tt ident}\, (AFL. automation. shared. Server Discovery. Service Browser
                                                                                                                                                                                                                                                                          method), 77, 738, 748
                                                                                                                                                                                                                                     \verb"init_models"() (AFL. automation. shared. Diffraction Labeler. Diffra
                                     property), 795
\verb|import_name| (AFL. automation. APIS erver. APIS erver. Flask
                                                                                                                                                                                                                                                                           method), 81, 761, 769
                                     attribute), 130
                                                                                                                                                                                                                                     init_plots() (AFL.automation.shared.DatasetWidget.DatasetWidget_Vie
in_bounds()
                                                                  (AFL.automation.prepare.MassBalance
                                                                                                                                                                                                                                                                           method), 80, 754, 758
                                     method), 386
                                                                                                                                                                                                                                    init_remote_connection()
indent (AFL. automation. prepare. Deck Builder Widget. Checkbox
                                                                                                                                                                                                                                                                           (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                                                                                               method),
                                      attribute), 410
indent(AFL. automation. prepare. PrepareWidget. Checkboxinitialize\_plots()(AFL. automation. shared. DatasetWidget. DatasetWi
                                      attribute), 502
                                                                                                                                                                                                                                                                          method), 79, 751, 757
indent(AFL.automation.prepare.SampleSeriesWidget.Checkbyect_url_defaults()
                                                                                                                                                                                                                                                                           (AFL.automation.APIServer.APIServer.Flask
                                      attribute), 576
indent (AFL.automation.prepare.SweepBuilderWidget.Checkbox
                                                                                                                                                                                                                                                                           method), 128
                                      attribute), 661
                                                                                                                                                                                                                                     instance\_path(AFL.automation.APIServer.APIServer.Flask
index (AFL. automation. prepare. Deck Builder Widget. Dropdown
                                                                                                                                                                                                                                                                           attribute), 113
                                                                                                                                                                                                                                     \verb|interface_index| (AFL. automation. APIS erver. APIS erver. Service Info
                                      attribute), 425
index (AFL.automation.prepare.PrepareWidget.Dropdown
                                                                                                                                                                                                                                                                           attribute), 149
                                      attribute), 514
                                                                                                                                                                                                                                     \verb"interface_index" (AFL. automation. shared. Server Discovery. Async Service Service
index() (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                                                                                                                                                                           attribute), 781
                                      method), 18, 100, 158
                                                                                                                                                                                                                                     interface_index(AFL.automation.shared.ServerDiscovery.ServiceInfo
index_new() (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                                                                                                                                                                           attribute), 799
                                     method), 18, 100, 158
                                                                                                                                                                                                                                    IntText (class in AFL.automation.prepare.SampleSeriesWidget),
infrequent_categories_
                                      (AFL.automation.shared.DataLabelerWidget.OrdinalFahadleroken_loader()
                                      attribute), 742
                                                                                                                                                                                                                                                                           (AFL.automation.APIServer.APIServer.JWTManager
infrequent_categories_
                                                                                                                                                                                                                                                                           method), 139
                                      (AFL.automation.shared.DataLabelerWidget.Ordinalenseletransform()
                                                                                                                                                                                                                                                                           (AFL. automation. shared. Data Labeler Widget. Ordinal Encoder
                                     property), 746
infrequent_categories_
                                                                                                                                                                                                                                                                          method), 745
                                      (AFL.automation.shared.DiffractionLabeler.OrdinianEncodertransform()
                                     attribute), 763
                                                                                                                                                                                                                                                                           (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
infrequent_categories_
                                                                                                                                                                                                                                                                           method), 766
                                     (AFL. automation. shared. Diffraction Labeler. Ordining {\it Encodings} 
                                                                                                                                                                                                                                                                            (AFL.automation.APIServer.APIServer.ServiceInfo
                                     property), 767
init() (AFL.automation.APIServer.APIServer
                                                                                                                                                                                                                                                                           method), 151
                                                                                                                                                                                                                                     ip_addresses_by_version()
                                     method), 19, 100, 159
init_app() (AFL.automation.APIServer.APIServer.CORS
                                                                                                                                                                                                                                                                           (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                                      method), 103
                                                                                                                                                                                                                                                                           method), 783
init_app() (AFL.automation.APIServer.APIServer.JWTMapageddresses_by_version()
                                                                                                                                                                                                                                                                           (AFL. automation. shared. Server Discovery. Service Info
init_buttons() (AFL.automation.shared.DatasetWidget.DatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidge
                                      method), 80, 754, 758
                                                                                                                                                                                                                                    IPVersion (class in AFL.automation.APIServer.APIServer),
```

| 135  | 87, 812   |                   |                          |
|--|---|-------------------|--------------------------|
| IPVersion (class in AFL.automation.shared.ServerDiscoventys), mola 788   | rity() (<br>AFL.automation.prepa                  | `                 | module                   |
| is_absolute()(AFL.automation.instrument.SeabreezeUVVis_Rolla   |   | •                 | module                   |
| method), 214   | AFL.automation.share                              | d.units), 87, 812 |                          |
| is_alive() (AFL.automation.APIServer.APIServer.QueueDserworm method), 143  | t() (AFL.automation.in<br>method), 213            | nstrument.Seabre  | ezeUVVis.Path            |
| is_alive() (AFL.automation.APIServer.QueueDaemon.QicsuaDelae   | * *   | nation.instrument | .SeabreezeUVVis.Path     |
| method), 191   | method), 214                                      |                   |                          |
| is_alive() (AFL.automation.EpicsADLiveProcess.ReduceTearnese   | * *   | on.instrument.Sec | abreezeUVVis.Path        |
| method), 825   | method), 214                                      |                   |                          |
| is_alive() (AFL.automation.loading.LoadStopperDriver.SaysePa   | Hilmg Telchedd                                    | (in               | module                   |
| method), 254   | AFL.automation.APISe                              | erver.QueueDaen   | ion),                    |
| $is\_alive()$ (AFL. automation. loading. Load Stopper Driver. Stop Loading. Load Stop Loading. Load Stop Loading. Load Stop Load St         | CB√1  |                   |                          |
| method), 257 is_seri   | alized()  | (in               | module                   |
| $is\_alive()$ (AFL.automation.loading.LoadStopperDriver.StopLoad   | CAFAL.automation.share                            | d.serialization), | 86,                      |
| method), 260   | 811   |                   |                          |
| is_alive() (AFL.automation.loading.Sensor.DummySensairs_serv<br>method), 335   | er_live() (AFL.auton<br>method), 18, 99, 158      | nation.APIServer  | APIServer.APIServer      |
| is_alive() (AFL.automation.loading.Sensor.DummySensairs_sock   |   | instrument.Seabr  | eezeUVVis.Path           |
| method), 338   | method), 213                                      |                   |                          |
| is_alive() (AFL.automation.loading.SensorCallbackThreiud_Senla   | t <b>.C</b> d <b>.H.F.d.c.k:Tihorerad</b> ion.pre | epare.Component   | t.Component              |
| method), 343   | property), 64, 396, 399                           | )                 | •                        |
| is_alive() (AFL.automation.loading.SensorCallbackThreius_Simple  | tEhreAFild@Bomation.p                             | repare.Solute pr  | operty),                 |
| method), 346   | 389   |                   |                          |
| is_alive() (AFL.automation.loading.SensorCallbackThreius_StopLanethod), 349  | <b>tæ</b> d <b>CAH</b> AL.automation.pr<br>393    | epare.Solvent pr  | operty),                 |
| is_alive() (AFL.automation.loading.SensorCallbackThreist_Stopk   | endCBFL.automation.p                              | repare.Componer   | nt.Component             |
| method), 352   | property), 64, 396, 399                           |                   | •                        |
| is_alive()(AFL.automation.loading.SensorPollingThread.Sesoln   |   |                   | operty),                 |
| method), 358   | 389   |                   |                          |
| is_alive() (AFL.automation.shared.ServerDiscovery.RunTbresolv  | ent (AFL.automation                               | .prepare.Solvent  | prop-                    |
| method), 790   | erty), 393  |                   |                          |
| is_alive()(AFL.automation.shared.ServerDiscovery.ServiseBynds  | ienk() (AFL.automation                            | n.instrument.Seal | reezeUVVis.Path          |
| method), 795   | method), 213                                      |                   |                          |
| is_block_device() (AFL.automation.instrument.Seabreeisd_VVVIu  | rRes(i) (is                                       | n                 | module                   |
| method), 213   | AFL.automation.prepa                              |                   |                          |
| $\verb is_char_device()  (AFL. automation. instrument. Seabreez   \verb als   V   V   od fine the content of the conte$ | mth() (in module AFL.a                            | utomation.shared  | d.units),                |
| method), 213   | 87, 812   |                   |                          |
|  | n() (AFL.automation.A                             | .PIServer.APISer  | ver.QueueDaemon          |
| AFL.automation.prepare.factory), 712   | method), 143                                      |                   |                          |
|  |   | .PIServer.QueueI  | Daemon.QueueDaemon       |
| AFL.automation.shared.units), 87, 812  | method), 191                                      |                   |                          |
|  | n() (AFL.automation.E                             | EpicsADLiveProce  | ess.ReduceDaemon.Reduc   |
| AFL.automation.shared.units), 87, 812  | method), 825                                      |                   |                          |
| is_dir() (AFL.automation.instrument.SeabreezeUVVis.PailsDaemo method), 213   | n() (AFL.automation.lo<br>method), 254            | oading.LoadStopp  | perDriver.SensorPollingT |
| is_fifo() (AFL.automation.instrument.SeabreezeUVVis.PiusDaemo  | , ·   | oading LoadStop   | perDriver.StopLoadCBv1   |
| method), 213   | method), 257                                      | ruung.20 uustepp  | ,e.B.,,,e.istop2eauc2,,1 |
| is_file() (AFL.automation.instrument.SeabreezeUVVis.PiusDaemo  | , ·   | oading.LoadSton   | perDriver,StopLoadCBv2   |
| method), 213   | method), 260                                      |                   |                          |
|  | n() (AFL.automation.le<br>method), 335            | oading.Sensor.Du  | mmySensor1               |
|  | n() (AFL.automation.le                            | oading.Sensor.Du  | mmySensor2               |
|  | · ·   | ~                 | •                        |

```
method), 338
                                                                                                             jinja_loader (AFL.automation.APIServer.APIServer.Flask
isDaemon() (AFL.automation.loading.SensorCallbackThread.SensorCadpleaty)Thisad
                 method), 343
                                                                                                             jinja_options (AFL.automation.APIServer.APIServer.Flask
isDaemon() (AFL.automation.loading.SensorCallbackThread.Simple\( \bar{a} \) hrebshteld \( \bar{C} \) B2
                 method), 346
                                                                                                             join() (AFL.automation.APIServer.APIServer.QueueDaemon
isDaemon() (AFL.automation.loading.SensorCallbackThread.StopLound(DBd), 144
                                                                                                             join() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
                 method), 349
isDaemon() (AFL.automation.loading.SensorCallbackThread.StopLoand(1)Bv12, 191
                 method), 352
                                                                                                             join() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDae
isDaemon() (AFL.automation.loading.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPolli
                 method), 358
                                                                                                             join() (AFL.automation.loading.LoadStopperDriver.SensorPollingThread
isDaemon() (AFL.automation.shared.ServerDiscovery.RunThread method), 255
                 method), 790
                                                                                                             join() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
isDaemon() (AFL.automation.shared.ServerDiscovery.ServiceBrows@method), 257
                                                                                                              join() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
                  method), 795
items() (AFL.automation.APIServer.Driver.PersistentConfig
                                                                                                                                method), 260
                  method), 171
                                                                                                             join() (AFL.automation.loading.Sensor.DummySensor1
items() (AFL.automation.APIServer.QueueDaemon.DataTrashcan method), 335
                                                                                                             join() (AFL.automation.loading.Sensor.DummySensor2
                 method), 189
items() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defauththlyt338
                 method), 280
                                                                                                             join() (AFL.automation.loading.SensorCallbackThread.SensorCallbackT
items() (AFL.automation.loading.PneumaticPressureSampleCell.defauthdid), 343
                                                                                                              join() (AFL.automation.loading.SensorCallbackThread.SimpleThreshold
                  method), 289
items() (AFL.automation.loading.PneumaticSampleCell.defaultdict method), 346
                  method), 299
                                                                                                             join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
items() (AFL.automation.loading.PushPullSelectorSampleCell.defauthtellood), 349
                                                                                                              join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
                  method), 315
items() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdiethod), 352
                 method), 326
                                                                                                             join() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
items() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.deficethtold); 358
                  method), 372
                                                                                                              join() (AFL.automation.shared.ServerDiscovery.RunThread
items()(AFL.automation.shared.DatasetWidget.defaultdict
                                                                                                                                method), 790
                  method), 756
                                                                                                             join() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
items() (AFL.automation.shared.PersistentConfig.MutableMapping method), 796
                  method), 772
                                                                                                             joinpath() (AFL.automation.instrument.SeabreezeUVVis.Path
items() (AFL.automation.shared.PersistentConfig.PersistentConfig method), 214
                 method), 775
                                                                                                                                    (AFL.automation.APIServer.APIServer.Flask
iter_blueprints() (AFL.automation.APIServer.APIServer.Flask attribute), 113
                  method), 119
                                                                                                             json\_decoder (AFL. automation. APIServer. APIServer. Flask
iterate_protocols() (AFL.automation.prepare.Deck
                                                                                                                               property), 112
                                                                                                             json_encoder(AFL.automation.APIServer.APIServer.Flask
                 method), 384
iterationid() (AFL.automation.APIServer.APIServer.MutableQueuproperty), 112
                 method), 141
                                                                                                             json_provider_class
iterationid() (AFL. automation. shared. Mutable Queue. Mutable QueAFL. automation. APIS erver. APIS erver. Flask
                  method), 82, 770, 771
                                                                                                                                attribute), 112
iterdir() (AFL.automation.instrument.SeabreezeUVVis.Pasonify()
                                                                                                                                                                                                         module
                                                                                                                                                                   (in
                  method), 211
                                                                                                                                AFL.automation.APIServer.APIServer), 92
                                                                                                             justify_content(AFL.automation.prepare.DeckBuilderWidget.Layout
J
                                                                                                                                attribute), 448
                                                                                                             justify_content(AFL.automation.prepare.PrepareWidget.Layout
jinja_env (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                attribute), 537
                 property), 115
\verb|jinja_environment| (AFL. automation. APIS erver. A
                                                                                                                                attribute), 620
                 attribute), 111
                                                                                                             justify_content(AFL.automation.prepare.SweepBuilderWidget.Layout
```

| attribute), 686  | keys (AFL.automation.prepare.PrepareWidget.VBox at-  |
|--|--|
| $\verb"justify_items" (AFL. automation. prepare. Deck Builder Williams") and the property of the $ | dget.Layoutribute), 561  |
| attribute), 448  | keys (AFL.automation.prepare.SampleSeriesWidget.Button   |
| justify_items(AFL.automation.prepare.PrepareWidget.  |  |
| attribute), 537  | keys (AFL.automation.prepare.SampleSeriesWidget.Checkbox   |
| justify_items(AFL.automation.prepare.SampleSeriesW   |  |
| attribute), 620  | keys (AFL.automation.prepare.SampleSeriesWidget.FloatText  |
|  |  |
| justify_items(AFL.automation.prepare.SweepBuilderW   | · ·  |
| attribute), 686  | keys (AFL.automation.prepare.SampleSeriesWidget.HBox   |
| <pre>jwt_required()</pre>  | attribute), 597  |
| AFL.automation.APIServer.APIServer), 93  | ${\tt keys}  (AFL. automation. prepare. Sample Series Widget. Int Text$  |
| JWTManager (class in AFL.automation.APIServer.APIServe   | er), attribute), 605   |
| 136  | keys (AFL.automation.prepare.SampleSeriesWidget.Label  |
|  | attribute), 613  |
| K  | keys (AFL.automation.prepare.SampleSeriesWidget.Layout   |
| key (AFL.automation.APIServer.APIServer.ServiceInfo  | attribute), 623  |
|  | keys (AFL.automation.prepare.SampleSeriesWidget.Text   |
| attribute), 149  |  |
| key (AFL.automation.instrument.SeabreezeUVVis.Eq at-   | attribute), 634  |
| tribute), 208  | keys (AFL.automation.prepare.SampleSeriesWidget.VBox   |
| key (AFL.automation.shared.ServerDiscovery.AsyncServic   |  |
| attribute), 781  | ${\bf keys}  (AFL. automation. prepare. Sweep Builder Widget. Button$  |
| ${\bf key} \ (AFL. automation. shared. Server Discovery. Service Info$   | attribute), 655  |
| attribute), 799  | keys (AFL.automation.prepare.SweepBuilderWidget.Checkbox   |
| keys (AFL.automation.prepare.DeckBuilderWidget.Button  | attribute), 663  |
| attribute), 404  | keys (AFL.automation.prepare.SweepBuilderWidget.HBox   |
| keys (AFL.automation.prepare.DeckBuilderWidget.Checkl  |  |
|  | keys (AFL.automation.prepare.SweepBuilderWidget.Label  |
| attribute), 412  |  |
| keys (AFL.automation.prepare.DeckBuilderWidget.Dropde  |  |
| attribute), 425  | keys (AFL.automation.prepare.SweepBuilderWidget.Layout   |
| keys (AFL.automation.prepare.DeckBuilderWidget.HBox  | attribute), 689  |
| attribute), 433  | ${\tt keys} \ (AFL. automation. prepare. Sweep Builder Widget. Text$   |
| keys (AFL.automation.prepare.DeckBuilderWidget.Label   | attribute), 699  |
| attribute), 441  | keys (AFL.automation.prepare.SweepBuilderWidget.VBox   |
| keys (AFL.automation.prepare.DeckBuilderWidget.Layout  | attribute), 707  |
| attribute), 451  | keys() (AFL.automation.APIServer.Driver.PersistentConfig   |
| keys (AFL.automation.prepare.DeckBuilderWidget.Text  | method), 171   |
| attribute), 459  | keys() (AFL.automation.APIServer.QueueDaemon.DataTrashcan  |
|  | · · · · ·  |
| keys (AFL.automation.prepare.DeckBuilderWidget.VBox  | method), 189   |
| attribute), 467  | keys() (AFL.automation.loading.OneSelectorBlowoutSampleCell.default  |
| keys (AFL.automation.prepare.PrepareWidget.Button at-  | method), 280   |
| tribute), 497  | $\textbf{keys()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. default and the property of the prope$ |
| keys (AFL.automation.prepare.PrepareWidget.Checkbox  | method), 289   |
| attribute), 504  | keys() (AFL.automation.loading.PneumaticSampleCell.defaultdict   |
| keys (AFL.automation.prepare.PrepareWidget.Dropdown  | method), 299   |
| attribute), 514  | ${\tt keys()} \ (AFL. automation. loading. Push Pull Selector Sample Cell. default dick and the property of the$     |
| keys (AFL.automation.prepare.PrepareWidget.HBox at-  | method), 315   |
| tribute), 522  | keys() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict   |
|  |  |
| keys (AFL.automation.prepare.PrepareWidget.Label at-   | method), 326   |
| tribute), 530  | keys() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.default  |
| keys (AFL.automation.prepare.PrepareWidget.Layout  | method), 372   |
| attribute), 540  | keys() (AFL.automation.shared.DatasetWidget.defaultdict  |
| keys (AFL.automation.prepare.PrepareWidget.Text at-  | method), 756   |
| tribute), 553  | ${\tt keys()} \ (AFL. automation. shared. Persistent Config. Mutable Mapping$  |
|  | method), 772   |
|  |  |

| keys() (AFL.automation.shared.Persis | stentConfig.Persistent <b>Caygig</b> t (AFL.automation.prepare.SampleSeriesWidget.FloatTex. |
|--------------------------------------|---|
| <i>method</i> ), 775                 | attribute), 589   |
|                                      | layout (AFL.automation.prepare.SampleSeriesWidget.HBox                                      |
|                                      | attribute) 597  |

- label (AFL.automation.prepare.DeckBuilderWidget.Dropd and (AFL.automation.prepare.SampleSeriesWidget.IntText attribute), 425 attribute), 605
- label (AFL.automation.prepare.PrepareWidget.Dropdown layout (AFL.automation.prepare.SampleSeriesWidget.Label attribute), 514 attribute), 613
- Label (class in AFL.automation.prepare.DeckBuilderWidger), ayout (AFL.automation.prepare.SampleSeriesWidget.Text 437
- Label (class in AFL.automation.prepare.PrepareWidget), 1ayout (AFL.automation.prepare.SampleSeriesWidget.VBox attribute), 642
- Label (class in AFL.automation.prepare.SampleSeriesWidger)yout (AFL.automation.prepare.SweepBuilderWidget.Button 609
- Label (class in AFL.automation.prepare.SweepBuilderWidglepyout (AFL.automation.prepare.SweepBuilderWidget.Checkbox attribute), 663
- label() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.HBox method), 77, 741, 748 attribute), 671
- label() (AFL.automation.shared.DiffractionLabeler.Diffrae ANDELE L.automation.prepare.SweepBuilderWidget.Label method), 81, 760, 768 attribute), 679
- latch\_shaker() (AFL.automation.prepare.Dummy\_OT2\_banker\_Ota\_banker
- layout (AFL.automation.prepare.DeckBuilderWidget.Buttohayout (AFL.automation.prepare.SweepBuilderWidget.VBox attribute), 404 attribute), 707
- layout (AFL.automation.prepare.DeckBuilderWidget.CheckBoxout (class in AFL.automation.prepare.DeckBuilderWidget), attribute), 412 445
- layout (AFL.automation.prepare.DeckBuilderWidget.DropWaut (class in AFL.automation.prepare.PrepareWidget), attribute), 425 534
- layout (AFL.automation.prepare.DeckBuilderWidget.HBoxLayout (class in AFL.automation.prepare.SampleSeriesWidget), attribute), 433 617
- layout (AFL.automation.prepare.DeckBuilderWidget.Labe|Layout (class in AFL.automation.prepare.SweepBuilderWidget), attribute), 441 683
- layout (AFL.automation.prepare.DeckBuilderWidget.Text lchmod() (AFL.automation.instrument.SeabreezeUVVis.Path attribute), 459 method), 212
- layout (AFL.automation.prepare.DeckBuilderWidget.VBoxleft (AFL.automation.prepare.DeckBuilderWidget.Layout attribute), 467 attribute), 448
- layout (AFL.automation.prepare.PrepareWidget.Button left (AFL.automation.prepare.PrepareWidget.Layout attribute), 497 (AFL.automation.prepare.PrepareWidget.Layout attribute), 537
- ${\bf layout}~(AFL. automation. prepare. Prepare Widget. Checkbox {\bf left}~(AFL. automation. prepare. Sample Series Widget. Layout~attribute),~620$
- layout (AFL.automation.prepare.PrepareWidget.Dropdown eft (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 514 attribute), 686
- layout (AFL.automation.prepare.PrepareWidget.HBox attribute), 522 link\_to() (AFL.automation.instrument.SeabreezeUVVis.Path method), 213
- layout (AFL.automation.prepare.PrepareWidget.Label listeners (AFL.automation.APIServer.APIServer.Zeroconf attribute), 530 property), 154
- layout (AFL.automation.prepare.PrepareWidget.Text attribute), 553

  listeners (AFL.automation.shared.ServerDiscovery.Zeroconf property), 805
- layout (AFL.automation.prepare.PrepareWidget.VBox listify() (in module attribute), 561

  AFL.automation.APIServer.APIServer), 94
- layout (AFL.automation.prepare.SampleSeriesWidget.Button stify() (in module attribute), 570 AFL.automation.APIServer.Driver), 166

| 173  |   | method), 277   |
|--|---|--|
| listify() (in                                      |   | loadSample() (AFL.automation.loading.PneumaticPressureSampleCell.P                       |
| AFL.automation.APIServer.Dummy(                    | OT2Driver),                             | method), 43, 286, 291  |
| 180  |   | loadSample() (AFL.automation.loading.PneumaticPressureSampleCell.S                       |
| listify() (in                                      | module                                  | method), 288   |
| •  | een_Driver),                            | , loadSample() (AFL.automation.loading.PneumaticSampleCell.Pneumatic                     |
| 219  |   | method), 45, 296, 300  |
| listify() (in module AFL.automation.prepa<br>713   | re.factory),                            | loadSample() (AFL.automation.loading.PneumaticSampleCell.SampleCemethod), 298            |
| listify() (in<br>AFL.automation.sample.CastingServ | module<br>ver).                         | loadSample() (AFL.automation.loading.PushPullSelectorSampleCell.Pusmethod), 48, 312, 317 |
| 718  | ,,                                      | loadSample() (AFL.automation.loading.PushPullSelectorSampleCell.Sam                      |
| listify() (in module AFL.automation.share          | d.utilities).                           | method), 313   |
| 88, 813, 814                                       | ,,                                      | loadSample() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX                        |
| load_cb() (AFL.automation.prepare.DeckBu           | ilderWidget.)                           |  |
| method), 65, 420, 471                              |   | loadSample() (AFL.automation.loading.RSoXSSolutionSampleCell.Samp                        |
| load_cb() (AFL.automation.prepare.Prepare          | Widget.Deck                             |  |
| method), 509                                       | ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, | loadSample() (AFL.automation.loading.SampleCell.SampleCell                               |
| load_cb() (AFL.automation.prepare.Prepare          | Widget Stock                            |  |
| method), 548                                       | .,,,,,,                                 | loadSample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell                        |
| load_cb() (AFL.automation.prepare.StockBu          | ilderWidget.                            |  |
| method), 72, 648, 650                              | rece // receers                         | loadSample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell                        |
| load_from_cache() (AFL.automation.APISe            | rver APISer                             |  |
| method), 151                                       | . ,                                     | LoadStopperDriver (class in  |
|  | d.ServerDisc                            | overy.Asyn <b>AFelrxiuednfa</b> tion.loading.LoadStopperDriver),                         |
| method), 783                                       |   | 39, 249, 261   |
|  | d.ServerDisc                            | olory(AdriviauInfnation.prepare.Component.ParseException                                 |
| method), 801                                       |   | attribute), 398  |
|  | ADLiveProce                             | essoGl(AvHICdiietotmation.prepare.DeckBuilderWidget.Button                               |
| method), 27, 822, 823                              |   | attribute), 404  |
|  | re.Dummy C                              | OTðgIQATVErdDtommyic@TpTepDniveDeckBuilderWidget.Checkbox                                |
| method), 67, 477, 479                              | 7—                                      | attribute), 412  |
|  | re.OT2Clien                             | t. <b>DECAFAnt</b> uutomation.prepare.DeckBuilderWidget.Dropdown                         |
| method), 67, 485, 487                              |   | attribute), 425  |
|  | e.CastingSer                            | rdengOTACliemutomation.prepare.DeckBuilderWidget.HBox                                    |
| method), 729                                       | Ö                                       | attribute), 433  |
|  | iveProcess.C                            | C <b>liog</b> t. <b>ChiFh</b> tautomation.prepare.DeckBuilderWidget.Label                |
| method), 27, 822, 823                              |   | attribute), 441  |
|  | Oummy OT2                               | _ <b>Doi</b> y& <b>A. Ph.mmt</b> ym@Ti&n. <b>preyea</b> re. DeckBuilder Widget. Layout   |
| method), 66, 477, 479                              | <i>y</i> = -                            | attribute), 452  |
|  | T2Client.OT                             | <b>AGb</b> ent(AFL.automation.prepare.DeckBuilderWidget.Text                             |
| method), 67, 485, 487                              |   | attribute), 460  |
|  | astingServer.                           | ATECN ANT. automation.prepare.DeckBuilderWidget.VBox                                     |
| method), 729                                       |   | attribute), 467  |
| LoaderCommunication (class                         | in                                      | log (AFL.automation.prepare.PrepareWidget.Button at-                                     |
| AFL.automation.loading.SensorCall                  |   |  |
| 56, 341, 355                                       | ,                                       | log (AFL.automation.prepare.PrepareWidget.Checkbox                                       |
| loadSample() (AFL.automation.APIServer.D           | ummyDriver                              |  |
| method), 23, 178, 180                              | <i>y</i>                                | log (AFL.automation.prepare.PrepareWidget.Dropdown                                       |
| loadSample() (AFL.automation.APIServer.D           | ummyOT2D                                |  |
| method), 24, 184, 186                              | ,                                       | log (AFL.automation.prepare.PrepareWidget.HBox at-                                       |
|  | SelectorBlow                            | voutSampletCidlut@neSelectorBlowoutSampleCell  |
| method), 273                                       |   | log (AFL.automation.prepare.PrepareWidget.Label at-                                      |
|  | SelectorRlow                            | voutSampleGiHlufEvoSellectorBlowoutSampleCell  |

| Log  | (AFL.automation.prepare.PrepareWidget.Layout attribute), 541           | log_exc           | eption_warning() (AFL.automation.APIServer.APIServer.Zeroconf                  |
|------|--|-------------------|--|
| Log  | (AFL. automation. prepare. Prepare Widget. Text                        |                   | class method), 157   |
|      |  | log_exc           | eption_warning()   |
| Log  | (AFL.automation.prepare.PrepareWidget.VBox at-                         |                   | (AFL.automation.shared.ServerDiscovery.Zeroconf                                |
|      | tribute), 561  |                   | class method), 808   |
| Log( | (AFL.automation.prepare.SampleSeriesWidget.Button attribute), 570      | log_war           | ning_once() (AFL.automation.APIServer.APIServer.Zeroconf class method), 157    |
| Log( | (AFL.automation.prepare.SampleSeriesWidget.Checkbo<br>attribute), 579  | o <b>k</b> og_war | ning_once() (AFL.automation.shared.ServerDiscovery.Zeroconf class method), 808 |
| Log( | (AFL.automation.prepare.SampleSeriesWidget.FloatTe.<br>attribute), 590 | xlrogged_         | in() (AFL.automation.APIServer.Client.Client<br>method), 20, 161, 164          |
| Log  | (AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 597        | logged_           | in() (AFL.automation.EpicsADLiveProcess.Client.Client method), 27, 822, 823    |
| Log( | (AFL.automation.prepare.SampleSeriesWidget.IntText attribute), 605     | logged_           | in() (AFL.automation.loading.LoadStopperDriver.Client method), 246             |
| Log  |  | logged_           | in() (AFL.automation.prepare.DeckBuilderWidget.Client method), 418             |
| Log( | (AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 624      | logged_           |  |
| Log  | (AFL.automation.prepare.SampleSeriesWidget.Text attribute), 635        | logged_           | in() (AFL.automation.prepare.OT2Client.OT2Client method), 485                  |
| Log  | (AFL.automation.prepare.SampleSeriesWidget.VBox attribute), 642        | logged_           | in() (AFL.automation.prepare.SampleSeriesWidget.Client method), 584            |
| Log( | (AFL.automation.prepare.SweepBuilderWidget.Button attribute), 655      | logged_           | in() (AFL.automation.sample.CastingServer.Client method), 723                  |
| Log( | (AFL.automation.prepare.SweepBuilderWidget.Checkbo<br>attribute), 663  | ∂lxogged_         | in() (AFL.automation.sample.CastingServer.OT2Client method), 729               |
| Log( | (AFL.automation.prepare.SweepBuilderWidget.HBox attribute), 671        | logger            | (AFL.automation.APIServer.APIServer.Flask property), 115                       |
| Log( | (AFL.automation.prepare.SweepBuilderWidget.Label attribute), 679       | LoggerF           | ilter (class in<br>AFL.automation.APIServer.APIServer), 141                    |
| Log( | (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 690      | LoggerF           | ilter (class in AFL.automation.APIServer.LoggerFilter),                        |
| Log  | (AFL.automation.prepare.SweepBuilderWidget.Text                        |                   | 25, 186, 187   |
|      |  | login()           | (AFL.automation.APIServer.APIServer.APIServer                                  |
| Log  | (AFL.automation.prepare.SweepBuilderWidget.VBox                        |                   | method), 19, 101, 159  |
|      | attribute), 707  | login()           | (AFL.automation.APIServer.Client.Client  |
| Log_ | $\_\mathtt{event}()\ (AFL. automation. sample. Casting Server. Cast$   | tingServei        | method), 20, 161, 164  |
|      | method), 89, 720, 730  | login()           | (AFL.automation.EpicsADLiveProcess.Client.Client                               |
| Log_ | $\_$ exception() (AFL.automation.APIServer.APIServer.                  | .Flask            | method), 27, 822, 823  |
|      | method), 123   | login()           | (AFL.automation.loading.LoadStopperDriver.Client                               |
| Log_ | _exception_debug()   |                   | method), 246   |
|      | (AFL.automation.APIServer.APIServer.Zeroconf                           | login()           | (AFL.automation.prepare.DeckBuilderWidget.Client                               |
|      | class method), 157   |                   | method), 418   |
| Log_ |  | login()           |  |
| 5-   | (AFL.automation.shared.ServerDiscovery.Zeroco                          | _                 | method), 481   |
|      | · · · · · · · · · · · · · · · · · · ·                                  |                   | (AFL.automation.prepare.OT2Client.OT2Client                                    |
| Loa  | _exception_once()  | 5 ()              | method), 485   |
| 9-   |  | login()           | (AFL.automation.prepare.SampleSeriesWidget.Client                              |
|      | class method), 157   | - 5 ()            | method), 584   |
| Loa  |  | login()           | (AFL.automation.sample.CastingServer.Client                                    |
| 3-   | (AFL.automation.shared.ServerDiscovery.Zeroco.                         |                   | method), 723   |
|      | · · · · · · · · · · · · · · · · · · ·                                  |                   | (AFL.automation.sample.CastingServer.OT2Client                                 |
|      |  |                   |  |

```
method), 729
                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
login_test() (AFL.automation.APIServer.APIServer
                                                                                                                                                                                                                                                                                                                                                         method), 71, 630, 647
                                                method), 19, 101, 159
                                                                                                                                                                                                                                                                                                        make_pipette_params()
lstat() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
                                                 method), 212
                                                                                                                                                                                                                                                                                                                                                          method), 71, 630, 647
                                                                                                                                                                                                                                                                                                        make_protocol()
                                                                                                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.prepare.Deck
M
                                                                                                                                                                                                                                                                                                                                                          method), 384
\verb|make_aborter()| (AFL. automation. APIS erver. APIS erver. Final \verb|ke_protocol()| (AFL. automation. prepare. Prepare Widget. Sample Series and Protocol() (AFL. automation. prepare Widget. Sample Series and Protocol()) (AFL. automation. prepare Widget.
                                                                                                                                                                                                                                                                                                                                                          method), 546
                                                method), 115
\verb|make_align_script()| (AFL. automation. prepare. Deck | \verb|make_protocol()| (AFL. automation. prepare. Sample Series Widget. Sampl
                                                                                                                                                                                                                                                                                                                                                          method), 70, 628, 646
                                                  method), 384
make_all_labels() (AFL.automation.prepare.PrepareWilledisScripspessesWildEL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                                                                                           method), 12\overline{5}
                                                 method), 547
make_all_labels()(AFL.automation.prepare.SampleSeriesWadget!!RAMPRESEFRESWidget
                                                                                                                                                                                                                                                                                                                                                           (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         method),
                                                 method), 70, 629, 646
                                                                                                                                                                                                                                                                                                                                                           384
make_all_labels_cb()
                                                 (AFL.automation.prepare.PrepareWidget.SampleSerkesWAGEipt()
                                                                                                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                           method), 384
                                                method), 547
                                                                                                                                                                                                                                                                                                        make_shell_context()
make_all_labels_cb()
                                                (AFL. automation. prepare. Sample Series Widget. Sample Series Widget automation. API Server. API Server. Flask automation and the sample Series Widget automation and the sample Series will be sample Series with the sam
                                                                                                                                                                                                                                                                                                                                                          method), 117
                                                method), 70, 629, 646
make_binary_plot()(AFL.automation.prepare.SweepBuilleteVisiteOf.SweepBuilletevwutgenanjopwprepare.PrepareWidget.StockBuilderVisiteOf.SweepBuilletev
                                                                                                                                                                                                                                                                                                                                                          method), 548
                                                method), 73, 695, 711
                                                                                                                                                                                                                                                                                                        make_stock_cb() (AFL.automation.prepare.StockBuilderWidget.StockBuil
make_catch_protocol()
                                                  (AFL.automation.prepare.PrepareWidget.SampleSeriesWidgethod), 72, 648, 650
                                                                                                                                                                                                                                                                                                        make_stock_grid() (AFL.automation.prepare.SweepBuilderWidget.Swee
                                                method), 546
                                                                                                                                                                                                                                                                                                                                                           method), 73, 695, 711
make_catch_protocol()
                                                 (AFL. automation. prepare. Sample Series Widget. Sample Series Widget. Stock Builder W
                                                                                                                                                                                                                                                                                                                                                         method), 72, 649, 650
                                                method), 70, 628, 646
                                                                                                                                                                                                                                                                                                        make_target_component_masses()
make_component_grid()
                                                 (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Equtypeqtion.prepare.MassBalance
                                                                                                                                                                                                                                                                                                                                                          method), 386
                                                method), 71, 630, 647
make_config()(AFL.automation.APIServer.APIServer.Flamake_ternary_plot()
                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWid
                                                method), 115
                                                                                                                                                                                                                                                                                                                                                         method), 73, 695, 711
make_default_options_response()
                                                                                                                                                                                                                                                                                                        make_wellplate_locs()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 (in
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 module
                                                 (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                                                                                         AFL.automation.prepare), 381
                                                method), 124
make_grid_mask() (AFL.automation.prepare.MassBalanc@make_wellplate_locs()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 module
                                                                                                                                                                                                                                                                                                                                                          AFL.automation.prepare.utilities), 74, 717
                                                method), 386
                                                                                                                                                                                                                                                                                                        makeRegistar()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 module
make_locs() (in module AFL.automation.prepare), 381
                                                                                                                                                                                                                                                                                                                                                         AFL.automation.prepare.PrepType), 488
make_locs()
                                                                                                                                                                                                                                                       module
                                                                                                                                                        (in
                                                                                                                                                                                                                                                                                                        makeRegistar()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 module
                                                                                                                                                                                                                                                                                                                                                                                                                                                                            (in
                                                AFL.automation.prepare.utilities), 74, 717
                                                                                                                                                                                                                                                                                                                                                         AFL.automation.shared.utilities),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              813.
make_mass_balance() (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                          814
                                                method), 384
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 module
                                                                                                                                                                                                                                                                                                        makeRegistrar()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                              (in
make_mass_fraction_matrix()
                                                                                                                                                                                                                                                                                                                                                         AFL.automation.APIServer.Driver),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     21,
                                                 (AFL.automation.prepare.MassBalance
                                                                                                                                                                                                                                                                                                                                                           166, 171
                                                method), 386
                                                                                                                                                                                                                                                                                                        manage_service_info_to_list()
make_mixing_wells()
                                                 (AFL. automation. Prepare. PrepareWidget. Sample SeriesWidge \ref{prop:prop:seriesWidget}. APL. automation. API Server. Client. Server Discovery to the property of the prope
                                                                                                                                                                                                                                                                                                                                                           method), 163
                                                method), 547
                                                                                                                                                                                                                                                                                                        manage_service_info_to_list()
make_mixing_wells()
                                                  (AFL. automation. prepare. Sample Series Widget. Sample Series Widge \mu to mation. shared. Server Discovery. Server Discovery the sample Series Widge \mu to mation. Shared the sample Server Discovery the sample Series Widge \mu to mation. Shared the sample Series Widge \mu to mation. Shared the sample Series Widge \mu to mation the sample Series Widge \mu to mation. Shared the sample Series Widge \mu to mation the sample Series Widge \mu to mation the sample Series Widge \mu to mation. Shared the sample Series Widge \mu to mation the Series Widge has the sample Ser
                                                                                                                                                                                                                                                                                                                                                          method), 85, 792, 808
                                                method), 71, 629, 646
                                                                                                                                                                                                                                                                                                        margin (AFL.automation.prepare.DeckBuilderWidget.Layout
make_mixing_wells()
```

| attribute), 448  | attribute), 621   |
|--|---|
|  | max_width (AFL.automation.prepare.SweepBuilderWidget.Layout               |
| attribute), 537  | attribute), 687   |
|  | comeasure_out() (AFL.automation.prepare.factory.Solution                  |
| attribute), 620  | method), 716  |
| margin (AFL.automation.prepare.SweepBuilderWidget.Lag  |   |
| attribute), 686  | method), 392  |
| Markdown (class in AFL.automation.shared.widgetui),  | metadata (AFL.automation.shared.widgetui.Markdown                         |
| 817  | attribute), 818   |
| <pre>mask_serialized_objs()</pre>  | min_height (AFL.automation.prepare.DeckBuilderWidget.Layout               |
| (AFL.automation.APIServer.APIServer.QueueDa  |   |
| method), 144   | min_height (AFL.automation.prepare.PrepareWidget.Layout                   |
| mask_serialized_objs()   | attribute), 538   |
| _  | u <b>wine hwei</b> nght (AFL.automation.prepare.SampleSeriesWidget.Layout |
| method), 25, 191, 192  | attribute), 621   |
| mass (AFL.automation.prepare.Component.Component   | min_height (AFL.automation.prepare.SweepBuilderWidget.Layout              |
| property), 64, 396, 398  | attribute), 687   |
| mass (AFL.automation.prepare.factory.Solution prop-  | min_width(AFL.automation.prepare.DeckBuilderWidget.Layout                 |
| erty), 715   | attribute), 449   |
| mass (AFL.automation.prepare.Solute property), 390   | min_width(AFL.automation.prepare.PrepareWidget.Layout                     |
| mass (AFL.automation.prepare.Solution property), 391   | attribute), 538   |
| mass (AFL.automation.prepare.Solvent property), 393  | min_width (AFL.automation.prepare.SampleSeriesWidget.Layout               |
| mass_fraction (AFL.automation.prepare.factory.Solution   |   |
| property), 715   | min_width (AFL.automation.prepare.SweepBuilderWidget.Layout               |
| mass_fraction (AFL.automation.prepare.Solution   | attribute), 687   |
| property), 392   | MixingException, 86, 810  |
| mass_totals_component()  | mkdir() (AFL.automation.instrument.SeabreezeUVVis.Path                    |
| (AFL.automation.prepare.SampleSeries   | method), 212  |
| method), 388   | model_id(AFL.automation.prepare.DeckBuilderWidget.Button                  |
| mass_totals_stock()  | property), 404  |
| (AFL.automation.prepare.SampleSeries   | model_id(AFL.automation.prepare.DeckBuilderWidget.Checkbox                |
| method), 388   | property), 412  |
| MassBalance (class in AFL.automation.prepare), 384   | model_id(AFL.automation.prepare.DeckBuilderWidget.Dropdown                |
| match() (AFL.automation.instrument.SeabreezeUVVis.Par  |   |
| method), 214   | model_id(AFL.automation.prepare.DeckBuilderWidget.HBox                    |
| match_server_by_name()   | property), 433  |
| · · · · · · · · · · · · · · · · · · ·  | emodel_id(AFL.automation.prepare.DeckBuilderWidget.Label                  |
| method), 163   | property), 441  |
| match_server_by_name()   | model_id(AFL.automation.prepare.DeckBuilderWidget.Layout                  |
| (AFL.automation.shared.ServerDiscovery.Server  |   |
| method), 85, 792, 809  | model_id(AFL.automation.prepare.DeckBuilderWidget.Text                    |
| max_height (AFL.automation.prepare.DeckBuilderWidge  | * *   |
| attribute), 449  | model_id(AFL.automation.prepare.DeckBuilderWidget.VBox                    |
| max_height (AFL.automation.prepare.PrepareWidget.Lay   | · · · · · · · · · · · · · · · · · · ·                                     |
| attribute), 538  | model_id(AFL.automation.prepare.PrepareWidget.Button                      |
| max_height (AFL.automation.prepare.SampleSeriesWidge   |   |
| attribute), 621  | model_id(AFL.automation.prepare.PrepareWidget.Checkbox                    |
| max_height (AFL.automation.prepare.SweepBuilderWidge   |   |
| attribute), 687  | model_id(AFL.automation.prepare.PrepareWidget.Dropdown                    |
| max_width (AFL.automation.prepare.DeckBuilderWidget.   |   |
| attribute), 449  | model_id(AFL.automation.prepare.PrepareWidget.HBox                        |
| max_width (AFL.automation.prepare.PrepareWidget.Layo   |   |
| attribute), 538  | model_id(AFL.automation.prepare.PrepareWidget.Label                       |
| max_width (AFL.automation.prepare.SampleSeriesWidget   |   |
| man="1" to the control of the contro | 20.50m p. oper.15), eee   |

```
model_id(AFL.automation.prepare.PrepareWidget.Layout
                                                           26, 820
        property), 541
                                                      AFL.automation.EpicsADLiveProcess.Client,
model_id (AFL.automation.prepare.PrepareWidget.Text
                                                           26, 821
                                                      AFL.automation.EpicsADLiveProcess.ReduceDaemon,
        property), 554
model\_id(AFL.automation.prepare.PrepareWidget.VBox
                                                           27, 823
        property), 561
                                                      AFL.automation.instrument, 28, 193
model_id(AFL.automation.prepare.SampleSeriesWidget.ButtonAFL.automation.instrument.DummySAS,
                                                                                                28.
        property), 571
model_id(AFL.automation.prepare.SampleSeriesWidget.ChecklAFL.automation.instrument.FileCamera, 29,
                                                           198
        property), 579
model_id(AFL.automation.prepare.SampleSeriesWidget.FloatTAPL.automation.instrument.I22SAXS, 29, 199
                                                      AFL.automation.instrument.NetworkCamera,
        property), 590
model_id(AFL.automation.prepare.SampleSeriesWidget.HBox
                                                           29, 204
        property), 597
                                                      AFL.automation.instrument.SeabreezeUVVis,
{\tt model\_id}\ (AFL. automation. prepare. Sample Series Widget. Int Text
                                                           30, 205
        property), 606
                                                      AFL.automation.instrument.SpecScreen_Driver,
                                                           31, 219
model_id(AFL.automation.prepare.SampleSeriesWidget.Label
        property), 614
                                                      AFL.automation.loading, 32, 224
model_id(AFL.automation.prepare.SampleSeriesWidget.LayoutAFL.automation.loading.CetoniMultiPosValve,
        property), 624
                                                           34, 226
model_id(AFL.automation.prepare.SampleSeriesWidget.Text AFL.automation.loading.ChemyxSyringePump,
        property), 635
                                                           34, 228
model_id(AFL.automation.prepare.SampleSeriesWidget.VBox AFL.automation.loading.DigitalOutPressureController,
                                                           36, 234
        property), 642
model_id(AFL.automation.prepare.SweepBuilderWidget.ButtonAFL.automation.loading.DoubleViciMultiposSelector,
        property), 655
                                                           37, 237
model_id(AFL.automation.prepare.SweepBuilderWidget.ChecklaExL.automation.loading.DummyPump, 38, 240
                                                      AFL.automation.loading.FlowSelector, 38,
        property), 663
model\_id(AFL.automation.prepare.SweepBuilderWidget.HBox)
                                                           243
        property), 671
                                                      AFL.automation.loading.LoadStopperDriver,
model_id(AFL.automation.prepare.SweepBuilderWidget.Label
                                                           39, 244
        property), 679
                                                      AFL.automation.loading.MultiChannelRelay,
model_id(AFL.automation.prepare.SweepBuilderWidget.Layout
                                                           40, 262
        property), 690
                                                      AFL.automation.loading.NE1kSyringePump,
model_id(AFL.automation.prepare.SweepBuilderWidget.Text
                                                      AFL.automation.loading.OneSelectorBlowoutSampleCell,
        property), 700
model_id(AFL.automation.prepare.SweepBuilderWidget.VBox
        property), 707
                                                      AFL.automation.loading.PneumaticPressureSampleCell,
module
                                                           42, 281
    AFL.automation, 15, 819
                                                      AFL.automation.loading.PneumaticSampleCell,
    AFL.automation.APIServer, 16, 90
    AFL.automation.APIServer.APIServer, 17, 91
                                                      AFL.automation.loading.PressureController,
    AFL.automation.APIServer.Client, 19, 159
                                                           45, 301
                                                      AFL.automation.loading.PressureControllerAsPump,
    AFL.automation.APIServer.Driver, 21, 165
    AFL.automation.APIServer.DummyDriver, 23,
                                                           46, 302
        173
                                                      AFL.automation.loading.PushPullSelectorSampleCell,
    AFL.automation.APIServer.DummyOT2Driver,
                                                           47, 305
                                                      AFL.automation.loading.RSoXSSolutionSampleCell,
        24, 180
    AFL.automation.APIServer.LoggerFilter,
                                                           49, 317
                                                      AFL.automation.loading.SainSmartRelay,
    AFL.automation.APIServer.QueueDaemon, 25,
                                                           50, 328
                                                      AFL.automation.loading.SampleCell, 52, 332
    AFL.automation.EpicsADLiveProcess, 26, 820
                                                      AFL.automation.loading.Sensor, 52, 333
    AFL.automation.EpicsADLiveProcess.AreaDetectorAHveautomation.loading.SensorCallbackThread,
```

| 53,341 AFL.automation.loading.SensorPollingThread 56,355                            | AFL.automation.shared.units, 87, 811 d, AFL.automation.shared.utilities, 87, 813 AFL.automation.shared.widgetui, 88, 814 |
|---|--|
| AFL.automation.loading.SerialDevice, 57, 359  | molarity (AFL.automation.prepare.factory.Solution  |
| AFL.automation.loading.SyringePump, 57, 360   | property), 716 molarity (AFL.automation.prepare.Solution property), 392  |
| AFL.automation.loading.Tubing, 58, 362  | moles (AFL.automation.prepare.Component.Component  |
| AFL.automation.loading.TwoSelectorBlowout   |  |
| 58, 363   | moles (AFL.automation.prepare.Solute property), 390  |
|   | ntwbd3 ፋሲFL.automation.prepare.Solvent property), 394  |
| 60, 374   | move() (AFL.automation.APIServer.APIServer.MutableQueue  |
| AFL.automation.loading.ViciMultiposSelecto  |  |
| 61, 377   | move() (AFL.automation.shared.MutableQueue.MutableQueue  |
| AFL automation prepare, 61, 380   | method), 82, 770, 771  |
| AFL.automation.prepare.Component, 63, 394 AFL.automation.prepare.DeckBuilderWidget, | move_item() (AFL.automation.APIServer.APIServer.APIServer  |
| 64, 399   | method), 19, 100, 159 move_item() (AFL.automation.APIServer.Client.Client  |
| AFL.automation.prepare.Dummy_OT2_Driver,  | method), 20, 162, 165  |
| 66, 472   | move_item() (AFL.automation.loading.LoadStopperDriver.Client   |
| AFL.automation.prepare.factory, 74, 712   | method), 246   |
| AFL.automation.prepare.OT2Client, 67, 480   | move_item() (AFL.automation.prepare.DeckBuilderWidget.Client   |
| AFL.automation.prepare.PrepareWidget, 68,   | method), 418   |
| 491   | move_item() (AFL.automation.prepare.OT2Client.Client   |
| AFL.automation.prepare.PrepType, 68, 488  | method), 482   |
| AFL.automation.prepare.SampleSeriesWidget, 70,565                                   | move_item() (AFL.automation.prepare.OT2Client.OT2Client method), 485   |
| 71, 647   | move_item() (AFL.automation.prepare.SampleSeriesWidget.Client method), 585   |
| 72, 650   | move_item() (AFL.automation.sample.CastingServer.Client method), 724   |
| AFL.automation.prepare.utilities, 74, 716<br>AFL.automation.sample, 89, 717         | move_item() (AFL.automation.sample.CastingServer.OT2Client method), 729  |
| AFL.automation.sample.CastingServer, 89, 717  | move_temp() (AFL.automation.sample_env.TemperatureDeck.Temperature method), 75, 735, 736                                 |
| AFL.automation.sample_env, 75, 731  | <pre>mpl_plot_to_bytes()</pre>   |
| AFL.automation.sample_env.TemperatureDeck, 75,731                                   | AFL.automation.shared.utilities), 88, 813, 814   |
| AFL.automation.shared, 76, 736  | ${\tt msg}(AFL. automation. prepare. Component. Parse Exception$   |
| AFL.automation.shared.DataLabelerWidget, 76,737                                     | attribute), 398<br>MultiChannelRelay (class in   |
| $ \begin{array}{cccccccccccccccccccccccccccccccccccc$                               | AFL.automation.loading.MultiChannelRelay),<br>40, 262, 263   |
| AFL.automation.shared.DiffractionLabeler, 80,759                                    | MultiChannelRelay (class in AFL.automation.loading.SainSmartRelay),  |
| AFL.automation.shared.exceptions, 86, 809   | 329  |
| AFL.automation.shared.MutableQueue, 82, 769   | MutableMapping (class in AFL.automation.shared.PersistentConfig),  |
| AFL.automation.shared.PersistentConfig,   | 771  |
| 83, 771   | MutableQueue (class in   |
| AFL.automation.shared.serialization, 86,  | AFL.automation.APIServer.APIServer), 141   |
| 810   | MutableQueue (class in   |
| AFL.automation.shared.ServerDiscovery, 84,776                                       | AFL.automation.shared.MutableQueue), 82, 769, 771  |

| N   | property), 825   |
|---|--|
| n_features_in_(AFL.automation.shared.DataLabelerWidget.Ord<br>attribute), 742                               | property), 233   |
| n_features_in_(AFL.automation.shared.DiffractionLabeller.Oxfinattribute), 763                               | property), 238   |
| property), 105  | _id(AFL.automation.loading.LoadStopperDriver.StopLoadCBv2 property), 261           |
| name (AFL.automation.APIServer.APIServer.Flask prop-  | _id(AFL.automation.loading.Sensor.DummySensor1 property), 335                      |
| name (AFL.automation.APIServer.APIServer.QueueDaemonnative property), 144                                   | property), 338   |
| name (AFL.automation.APIServer.APIServer.ServiceInfo native   | _id (AFL.automation.loading.SensorCallbackThread.SensorCallba<br>property), 344    |
| name (AFL.automation.APIServer.APIServer.SMTPHandlernative  | property), 346   |
| name (AFL.automation.APIServer.QueueDaemon.QueueDaemonive property), 191                                    | property), 349   |
| name (AFL.automation.EpicsADLiveProcess.ReduceDaemon.Reidue<br>property), 825                               | property), 552   |
| nronerty) 214   | _id (AFL.automation.loading.SensorPollingThread.SensorPollingT property), 358      |
| name (AFL.automation.loading.LoadStopperDriver.SensorPoliting In property), 255                             | property), 191   |
| name (AFL.automation.loading.LoadStopperDriver.StopLoadebyee property), 258                                 | property), 190   |
| name (AFL.automation.loading.LoadStopperDriver.StopLoadEbv2y property), 261                                 | AT L. automation. todating. TVLT KSyrtinger ump),                                  |
| name (AFL.automation.loading.Sensor.DummySensor1 property), 335 needs_                                      | 40, 263, 266 fresh_token_loader()  |
| name (AFL.automation.loading.Sensor.DummySensor2 property), 338   | (AFL.automation.APIServer.APIServer.JWTManager method), 139                        |
| name (AFL.automation.loading.SensorCallbackThread.Sens ปีคิตันกรับ<br>property), 344                        | AF L.automation.instrument.NetworkCamera),   |
| ${\it name (AFL. automation. loading. Sensor Callback Thread. Simple Thresproperty), 346} \\ {\it next\_b}$ | utton caliback()   |
| name (AFL.automation.loading.SensorCallbackThread.StopLoadCB property), 349                                 | memoa), 77, 740, 747   |
| name (AFL.automation.loading.SensorCallbackThread.StopPoddeB  | (AI L. automation. Sharea. Dataset wiaget. Dataset wiaget                          |
| property), 550  | utton_caliback()   |
| name (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo attribute), 783                                | (AFL.automation.shared.DiffractionLabeler.DiffractionLabeler method), 81, 760, 768 |
| name (AFL.automation.shared.ServerDiscovery.RunThread NotFou  | IIUEIIUI, 60, 610  |
| name (AFL.automation.shared.ServerDiscovery.ServiceBrowserify property), 796                                | metnoa), 154   |
| name (AFL.automation.shared.ServerDiscovery.ServiceInfo notify attribute), 801                              | <i>method</i> ), 805   |
| native_id (AFL.automation.APIServer.APIServer.QueueDaemon) property), 144                                   | metnoa), 405   |
| native_id(AFL.automation.APIServer.QueueDaemon.QueueDaen property), 191                                     | memoa), 415  |
| native_id(AFL.automation.EpicsADLiveProcess.ReduceDatify  | ReduceDaemon L. automation. prepare. Deckbullaer wiaget. Dropaow                   |

- method), 426 method), 690
- notify\_change() (AFL.automation.prepare.DeckBuilderWidgetHBahange() (AFL.automation.prepare.SweepBuilderWidget.Text method), 433 method), 700
- notify\_change() (AFL.automation.prepare.DeckBuilderWidgitflyabhlange() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 442 method), 707
- notify\_change() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 452
- notify\_change() (AFL.automation.prepare.DeckBuilderWidget-Lewit (AFL.automation.prepare.DeckBuilderWidget.Layout method), 460 attribute), 449
- notify\_change() (AFL.automation.prepare.DeckBuilderWidget\(\frac{VBP\}{2}\)t (AFL.automation.prepare.PrepareWidget.Layout method), 467

  attribute), 538
- notify\_change() (AFL.automation.prepare.PrepareWidgenButton\_fit (AFL.automation.prepare.SampleSeriesWidget.Layout method), 497 attribute), 621
- notify\_change() (AFL.automation.prepare.PrepareWidgetGbett (AFL.automation.prepare.SweepBuilderWidget.Layout method), 505 attribute), 687
- notify\_change() (AFL.automation.prepare.PrepareWidgetLayout method), 522 attribute), 538
- notify\_change() (AFL.automation.prepare.PrepareWidgetHztel\_position(AFL.automation.prepare.SampleSeriesWidget.Layout method), 531

  attribute), 621
- notify\_change() (AFL.automation.prepare.PrepareWidgetJzectt\_position(AFL.automation.prepare.SweepBuilderWidget.Layout method), 541 attribute), 687
- notify\_change() (AFL.automation.prepare.PrepareWidgetFeerve() (AFL.automation.prepare.DeckBuilderWidget.Button method), 554 method), 405
- notify\_change() (AFL.automation.prepare.PrepareWidget\Serve() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 561 method), 413

- notify\_change() (AFL.automation.prepare.SampleSeries WidgetvEloy (AFL.automation.prepare.DeckBuilderWidget.Label method), 590 method), 442
- notify\_change() (AFL.automation.prepare.SampleSeries Widget \( \frac{1}{2} \) \( \fr
- notify\_change() (AFL.automation.prepare.SampleSeries Widget Let Text Lautomation.prepare.DeckBuilderWidget.Text method), 460
- notify\_change() (AFL.automation.prepare.SampleSeries \ightharpoonup \ightharpoonu
- notify\_change() (AFL.automation.prepare.SampleSeries WidgetVeCy)(AFL.automation.prepare.PrepareWidget.Button method), 624 method), 497
- notify\_change() (AFL.automation.prepare.SampleSeries \( \) (AFL.automation.prepare.PrepareWidget.Checkbox method), 635 \( method), 505 \)
- notify\_change() (AFL.automation.prepare.SampleSeries WidgetveRox(AFL.automation.prepare.PrepareWidget.Dropdown method), 642 method), 515

- notify\_change() (AFL.automation.prepare.SweepBuilder Widget-Lautomation.prepare.PrepareWidget.Text method), 680 method), 554
- $\verb"notify_change" () \textit{ (AFL. automation. prepare. Sweep Builder Widget. Layout a)} \\$

- observe() (AFL.automation.prepare.PrepareWidget.VBox on\_msg() (AFL.automation.prepare.DeckBuilderWidget.Text method), 562 method), 460
- observe() (AFL.automation.prepare.SampleSeriesWidget.Bouttomsg() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 571 method), 468
- observe() (AFL.automation.prepare.SampleSeriesWidget.**Checkso**() (AFL.automation.prepare.PrepareWidget.Button method), 579 method), 497
- observe() (AFL.automation.prepare.SampleSeriesWidget.FokoantEgy() (AFL.automation.prepare.PrepareWidget.Checkbox method), 590 method), 505
- observe() (AFL.automation.prepare.SampleSeriesWidget.MBomsg() (AFL.automation.prepare.PrepareWidget.Dropdown method), 598 method), 515
- observe() (AFL.automation.prepare.SampleSeriesWidget.lomTemsg() (AFL.automation.prepare.PrepareWidget.HBox method), 606 method), 523
- observe() (AFL.automation.prepare.SampleSeriesWidget.kombedsg() (AFL.automation.prepare.PrepareWidget.Label method), 514 method), 531
- observe() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 624 method), 541
- observe() (AFL.automation.prepare.SampleSeriesWidget.Textmsg() (AFL.automation.prepare.PrepareWidget.Textmethod), 635 method), 554
- observe() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 643 (AFL.automation.prepare.PrepareWidget.VBox method), 562
- observe() (AFL.automation.prepare.SweepBuilderWidget.Bnttmsg() (AFL.automation.prepare.SampleSeriesWidget.Button method), 656 method), 571
- observe() (AFL.automation.prepare.SweepBuilderWidget.**6hewkly**(x) (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 664 method), 579
- observe() (AFL.automation.prepare.SweepBuilderWidget.bhBarsg() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 671 method), 590
- observe() (AFL.automation.prepare.SweepBuilderWidget.bmbrdsg() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 680 method), 598
- observe() (AFL.automation.prepare.SweepBuilderWidget.baymsg() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 690 method), 606
- observe() (AFL.automation.prepare.SweepBuilderWidget. Enginesg() (AFL.automation.prepare.SampleSeriesWidget.Label method), 700 method), 614
- observe() (AFL.automation.prepare.SweepBuilderWidget.VBomsg() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 708 method), 624
- on\_click() (AFL.automation.prepare.DeckBuilderWidget.Bnttmsg() (AFL.automation.prepare.SampleSeriesWidget.Text method), 402 method), 635
- on\_click() (AFL.automation.prepare.PrepareWidget.Buttom\_msg() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 495 method), 643
- on\_click() (AFL.automation.prepare.SampleSeriesWidgetcButtnsvg() (AFL.automation.prepare.SweepBuilderWidget.Button method), 568 method), 656
- on\_click() (AFL.automation.prepare.SweepBuilderWidgetCheckbox method), 653 method), 664
- on\_msg() (AFL.automation.prepare.DeckBuilderWidget.Buannmsg() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 405 method), 672
- on\_msg() (AFL.automation.prepare.DeckBuilderWidget.Chenklmsg() (AFL.automation.prepare.SweepBuilderWidget.Label method), 413 method), 680
- on\_msg() (AFL.automation.prepare.DeckBuilderWidget.Dropdmsg() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 426 method), 690
- on\_msg() (AFL.automation.prepare.DeckBuilderWidget.HBm\_msg() (AFL.automation.prepare.SweepBuilderWidget.Text method), 434 method), 700
- on\_msg() (AFL.automation.prepare.DeckBuilderWidget.Label\_msg() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 442 method), 708

| method), 163  | $on\_trait\_change()$ (AFL.automation.prepare.SampleSeriesWidget.Label)                               |
|---|---|
| on_service_state_change()                                   | method), 614  |
| (AFL.automation.shared.ServerDiscove method), 85, 792, 808  | ry.ServerDincanearyt_change() (AFL.automation.prepare.SampleSeriesWidget.Layor<br>method), 624        |
| on_submit()(AFL.automation.prepare.DeckBui                  | ilderWidgenNextrait_change()(AFL.automation.prepare.SampleSeriesWidget.Text                           |
| method), 457  | method), 635  |
| on_submit() (AFL.automation.prepare.Prepare<br>method), 551 | Widget.Texon_trait_change() (AFL.automation.prepare.SampleSeriesWidget.VBox<br>method), 643           |
|   | memod), 643<br>SeriesWidg <b>enTexr</b> ait_change() (AFL.automation.prepare.SweepBuilderWidget.Butto |
| method), 632  | method), 656  |
|   | uilderWidg <b>enT</b> &wait_change() (AFL.automation.prepare.SweepBuilderWidget.Chec                  |
| method), 697  | method), 664  |
| <pre>on_trait_change() (AFL.automation.prepare.l</pre>      | DeckBuild <b>onVidgai Budha</b> nge() (AFL.automation.prepare.SweepBuilderWidget.HBo                  |
| method), 405  | method), 672  |
| <pre>on_trait_change() (AFL.automation.prepare.)</pre>      | DeckBuild <b>&amp;M_ithgai.Chehlmge</b> () (AFL.automation.prepare.SweepBuilderWidget.Labe            |
| method), 413  | method), 680  |
| <pre>on_trait_change() (AFL.automation.prepare.)</pre>      | DeckBuild <b>onVidgai.Drohdwge()</b> (AFL.automation.prepare.SweepBuilderWidget.Layo                  |
| method), 426  | method), 690  |
| on_trait_change()(AFL.automation.prepare.l                  | DeckBuildan Widgai HBchange() (AFL.automation.prepare.SweepBuilderWidget.Text                         |
| method), 434  | method), 700  |
| on_trait_change()(AFL.automation.prepare.l                  | DeckBuild <b>onVithgai.Labhl</b> ange() (AFL.automation.prepare.SweepBuilderWidget.VBox               |
| method), 442  | method), 708  |
| on_trait_change()(AFL.automation.prepare.l                  |   |
| method), 452  | (AFL.automation.prepare.DeckBuilderWidget.Button  |
| on_trait_change()(AFL.automation.prepare.l                  |   |
| method), 460  | on_widget_constructed()   |
| **  | DeckBuilderWidget. ( <b>B&amp;L</b> .automation.prepare.DeckBuilderWidget.Checkbox                    |
| method), 468  | static method), 414   |
| on_trait_change() (AFL.automation.prepare.I                 |   |
| method), 497  | (AFL.automation.prepare.DeckBuilderWidget.Dropdown  |
| on_trait_change() (AFL.automation.prepare.I                 |   |
| method), 505  | on_widget_constructed()   |
|   | PrepareWidget.Drop(IAWh.automation.prepare.DeckBuilderWidget.HBox                                     |
| method), 515  | static method), 434   |
| on_trait_change() (AFL.automation.prepare.I                 |   |
| method), 523  | (AFL.automation.prepare.DeckBuilderWidget.Label   |
| on_trait_change() (AFL.automation.prepare.I                 |   |
| method), 531  | on_widget_constructed()   |
|   | PrepareWidget.LayouAFL.automation.prepare.DeckBuilderWidget.Layout                                    |
|   | static method), 453   |
| method), 541  |   |
| on_trait_change() (AFL.automation.prepare.i                 |   |
| method), 554  | (AFL.automation.prepare.DeckBuilderWidget.Text  |
| on_trait_change() (AFL.automation.prepare.                  |   |
| method), 562  | on_widget_constructed()   |
|   | SampleSeriesWidget. BAFibuutomation.prepare.DeckBuilderWidget.VBox                                    |
| method), 571  | static method), 468   |
| on_trait_change() (AFL.automation.prepare.S                 | 1 0 2   |
| method), 579  | (AFL.automation.prepare.PrepareWidget.Button  |
| on_trait_change() (AFL.automation.prepare.S                 |   |
| method), 590  | on_widget_constructed()   |
|   | SampleSeriesWidget. <b>{AF</b> dxautomation.prepare.PrepareWidget.Checkbox                            |
| method), 598  | static method), 506   |
| <pre>on_trait_change() (AFL.automation.prepare.S</pre>      |   |
| method), 606  | (AFL. automation. prepare. Prepare Widget. Dropdown   |

| static method), 516  | static method), 681   |
|--|---|
| on_widget_constructed()  | <pre>on_widget_constructed()</pre>                                  |
| (AFL. automation. prepare. Prepare Widget. HBox  | (AFL. automation. prepare. Sweep Builder Widget. Layout             |
| static method), 523  | static method), 691   |
| on_widget_constructed()  | on_widget_constructed()   |
| (AFL.automation.prepare.PrepareWidget.Label static method), 532  | (AFL.automation.prepare.SweepBuilderWidget.Text static method), 701 |
| on_widget_constructed()  | on_widget_constructed()   |
| (AFL.automation.prepare.PrepareWidget.Layout   | (AFL.automation.prepare.SweepBuilderWidget.VBox                     |
| static method), 542  | static method), 708   |
| on_widget_constructed()  | OneSelectorBlowoutSampleCell (class in                              |
| (AFL.automation.prepare.PrepareWidget.Text   | AFL.automation.loading.OneSelectorBlowoutSampleCell),               |
| static method), 555  | 41, 270, 280  |
| <pre>on_widget_constructed()</pre>   | open() (AFL.automation.instrument.SeabreezeUVVis.Path               |
| (AFL.automation.prepare.PrepareWidget.VBox   | method), 211  |
| static method), 562  | open() (AFL.automation.prepare.DeckBuilderWidget.Button             |
| <pre>on_widget_constructed()</pre>   | method), 406  |
| (AFL.automation.prepare.SampleSeriesWidget.Bu  | appen() (AFL.automation.prepare.DeckBuilderWidget.Checkbox          |
| static method), 572  | method), 414  |
| on_widget_constructed()  | open() (AFL.automation.prepare.DeckBuilderWidget.Dropdown           |
| (AFL. automation. prepare. Sample Series Widget. Compared to the substitution of the | heckbox method), 427  |
| static method), 580  | open() (AFL.automation.prepare.DeckBuilderWidget.HBox               |
| on_widget_constructed()  | method), 435  |
| (AFL. automation. prepare. Sample Series Widget. Floor and the series widget and the series with the series with the series with the series will be series with the series with the series will be series with the series with the series will be seri | op@n(t) (AFL.automation.prepare.DeckBuilderWidget.Label             |
| static method), 591  | method), 443  |
| on_widget_constructed()  | open() (AFL.automation.prepare.DeckBuilderWidget.Layout             |
| (AFL. automation. prepare. Sample Series Widget. H. automation. prepare and all the series widget. H. automation and all the series will be a series with the serie | Box method), 453  |
| static method), 598  | open() (AFL.automation.prepare.DeckBuilderWidget.Text               |
| on_widget_constructed()  | method), 461  |
|  | tapen() (AFL.automation.prepare.DeckBuilderWidget.VBox              |
| static method), 607  | method), 469  |
| on_widget_constructed()  | open() (AFL.automation.prepare.PrepareWidget.Button                 |
| (AFL. automation. prepare. Sample Series Widget. Logarithm 1995 and 1995 and 1995 are supported by the property of the prope |   |
| static method), 615  | open() (AFL.automation.prepare.PrepareWidget.Checkbox               |
| on_widget_constructed()  | method), 506  |
|  | appen() (AFL.automation.prepare.PrepareWidget.Dropdown              |
| static method), 625  | method), 516  |
| on_widget_constructed()  | open() (AFL.automation.prepare.PrepareWidget.HBox                   |
| (AFL.automation.prepare.SampleSeriesWidget.Te  |   |
| static method), 636  | open() (AFL.automation.prepare.PrepareWidget.Label                  |
| on_widget_constructed()  | method), 532  |
|  | Boppen() (AFL.automation.prepare.PrepareWidget.Layout               |
| static method), 643  | method), 542  |
| on_widget_constructed()  | open() (AFL.automation.prepare.PrepareWidget.Text                   |
| (AFL.automation.prepare.SweepBuilderWidget.B   |   |
| static method), 657  | open() (AFL.automation.prepare.PrepareWidget.VBox                   |
| on_widget_constructed()  | method), 563  |
|  | hapkbay (AFL.automation.prepare.SampleSeriesWidget.Button           |
| static method), 665  | method), 572  |
| on_widget_constructed()  | open() (AFL.automation.prepare.SampleSeriesWidget.Checkbox          |
| (AFL.automation.prepare.SweepBuilderWidget.H   |   |
| <pre>static method), 672 on_widget_constructed()</pre>   | open() (AFL.automation.prepare.SampleSeriesWidget.FloatText         |
|  | method), 591  |
| (АТ L.ашотаноп.ргераге.SweepBuнaerwlaget.La  | abpen() (AFL.automation.prepare.SampleSeriesWidget.HBox             |

| method), 599                               |                                     |            | AFL.automation.shared.DiffractionLabeler),                          |
|--|-------------------------------------|------------|---|
| open() (AFL.automation.prepar              | re.SampleSeriesWidget.IntT          | Text       | 762   |
| method), 607                               |                                     | OT2Clie    | nt (class in AFL.automation.prepare.OT2Client),                     |
| open() (AFL.automation.prepar              | re.SampleSeriesWidget.Lab           | pel        | 67, 483, 487  |
| method), 615                               |                                     |            | nt (class in AFL.automation.sample.CastingServer),                  |
| open() (AFL.automation.prepar              | re.SampleSeriesWidget.Lay           |            | 727   |
| method), 625                               |                                     |            | tl (AFL.automation.APIServer.APIServer.ServiceInfo                  |
| open() (AFL.automation.prepar              | re.SampleSeriesWidget.Tex           |            | attribute), 149   |
| method), 636                               |                                     |            | tl (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo          |
| open() (AFL.automation.prepar              | re.SampleSeriesWidget.VBe           |            | attribute), 781   |
| method), 644                               | 6 P. II. W. I. P.                   |            | tl (AFL.automation.shared.ServerDiscovery.ServiceInfo               |
| open() (AFL.automation.prepar              | re.SweepBuilderWidget.Bui           |            | attribute), 799   |
| method), 657                               | a pril wil a                        |            | w (AFL.automation.prepare.DeckBuilderWidget.Layout                  |
| open() (AFL.automation.prepar              | re.SweepBuilderWidget.Ch            |            | attribute), 449   |
| method), 665                               | a n ii wii .un                      |            | w (AFL.automation.prepare.PrepareWidget.Layout                      |
| open() (AFL.automation.prepar              | re.SweepBuilderWidget.HB            |            | attribute), 538   |
| method), 672                               | a pellari                           |            | w (AFL.automation.prepare.SampleSeriesWidget,Layout                 |
| open() (AFL.automation.prepar              | re.SweepBuilaerWiaget.Lat           |            | attribute), 621   |
| method), 681                               | C D.::11                            |            | w (AFL.automation.prepare.SweepBuilderWidget.Layout                 |
| open() (AFL.automation.prepar              | re.SweepBunaerwiaget.Lay            |            | attribute), 687   |
| method), 691                               | no Conson Devil don Wido at Ton     |            | (AFL.automation.instrument.SeabreezeUVVis.Path                      |
| open() (AFL.automation.prepar              | re.5weeр <b>ь</b> инаеr w taget.1ex | ı          | method), 211  |
| method), 701 open() (AFL.automation.prepar | ra Swaan Ruildar Widgat V.R.        | P          |   |
| method), 709                               | ге. эмеервинает үнаден. үв          |            |   |
| open_instance_resource()                   |                                     |            | _cmd() (AFL.automation.loading.UltimusVPressureController.Ul        |
| =  | Server.APIServer.Flask              |            | method), 60, 376, 377   |
| method), 116                               | gerver.m igerver.i task             | padding    | (AFL.automation.prepare.DeckBuilderWidget.Layout                    |
|  | ation APIServer APIServer           | · ELosh:   | attribute), 449<br>(AFL.automation.prepare.PrepareWidget.Layout     |
| method), 128                               | anomin iserverini iserver           |            |   |
|  | mation loading ChemyxSyr            | inaePama   | attribute), 538<br>(APPLYMG9MH46H9Prepare.SampleSeriesWidget.Layout |
| method), 36, 229, 233                      | nation.todaing.enemyxsyr            |            | attribute), 621   |
|  |                                     | ondawn     | (AFL.automation.prepare.SweepBuilderWidget.Layout                   |
| attribute), 427                            | ureizeenzumeer // magenz /          |            | attribute), 687   |
|  | are.PrenareWidget.Drondo            | Whron+ (   | AFL.automation.instrument.SeabreezeUVVis.Path                       |
| attribute), 516                            | aren repaire magens repair          |            | property), 214  |
|  | e.DeckBuilderWidget.Lavoi           | ltnaronts  | (AFL.automation.instrument.SeabreezeUVVis.Path                      |
| attribute), 449                            | ,                                   |            | property), 214  |
|  | re.PrepareWidget.Layout             |            | ell() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O              |
| attribute), 538                            | 1 0 7                               | _          | method), 66, 477, 479   |
|  | e.SampleSeriesWidget.Layo           | undarsed : | addresses() (AFL.automation.APIServer.APIServer.ServiceInfo         |
| attribute), 621                            | 1 0 7                               |            | method), 151  |
|  | e.SweepBuilderWidget.Layo           | Marsed:    | addresses() (AFL.automation.shared.ServerDiscovery.AsyncSer         |
| attribute), 687                            |                                     |            | method), 783  |
| ordinal_phase_labels()                     |                                     |            | addresses() (AFL.automation.shared.ServerDiscovery.ServiceIr        |
|  | ed.DataLabelerWidget.Dat            | aLabelerN  | 4949hod), 801   |
| method), 77, 738, 748                      |                                     |            | scoped_addresses()  |
| ordinal_phase_labels()                     |                                     | F          | (AFL.automation.APIServer.APIServer.ServiceInfo                     |
| (AFL.automation.shar                       | red.DiffractionLabeler.Diffr        | actionLab  | elerModel <sub>151</sub>  |
| method), 81, 761, 769                      | •                                   |            | scoped_addresses()  |
| OrdinalEncoder                             | (class in                           | _          | (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo             |
| AFL.automation.share                       | ed.DataLabelerWidget),              |            | method), 783  |
| 741  |                                     |            | scoped_addresses()  |
| OrdinalEncoder                             | (class in                           | . –        | (AFL.automation.shared.ServerDiscovery.ServiceInfo                  |

| method), 801   | placeholder (AFL.automation.prepare.SampleSeriesWidget.Text   |
|--|---|
| ParseException, 398  | attribute), 636   |
|  | placeholder(AFL.automation.prepare.SweepBuilderWidget.Label   |
| AFL.automation.loading.ChemyxSyringePump),   | attribute), 681   |
| 35, 228, 233   | placeholder (AFL.automation.prepare.SweepBuilderWidget.Text   |
| ${\tt parser\_element} \ (AFL. automation. prepare. Component. Part of the component of the c$  |   |
| attribute), 398  | plot_binary_cb() (AFL.automation.prepare.PrepareWidget.SweepBuilde  |
| ${\tt parts}  (AFL. automation. instrument. Seabreeze UVV is. Path$  | method), 548  |
| property), 214   | $\verb"plot_binary_cb"() (AFL. automation. prepare. Sweep Builder Widget. Sweep Builder Wid$  |
| patch() (AFL.automation.APIServer.APIServer.Flask  | method), 73, 693, 711   |
| method), 128   | plot_bounds() (AFL.automation.prepare.MassBalance   |
| ${\tt Path}(classinAFL. automation. instrument. Seabreeze UVV is a property of the contract $      |   |
| 209  | $\verb"plot_comp"()" (AFL. automation. shared. Dataset Widget. Dataset Widget\_View and the plot_comp") and the plot_comp () is a plot_comp ()$ |
| pause() (AFL.automation.APIServer.APIServer.APIServe   |   |
| method), 19, 100, 159  | <pre>plot_grid_mask() (AFL.automation.prepare.MassBalance</pre>   |
| pause() (AFL.automation.APIServer.Client.Client  | method), 386  |
| method), 20, 161, 164  | $\verb"plot_sas"()" (AFL. automation. shared. Dataset Widget\_Dataset Widget\_View")$   |
| $\verb"pause()" (AFL. automation. loading. Load Stopper Driver. Cliebase ()) and the property of $ | ent method), 80, 754, 758   |
| method), 246   | $\verb plot_ternary_cb()  (AFL. automation. prepare. Prepare Widget. Sweep Builde and the property of the propert$  |
| $\verb"pause()" (AFL. automation. prepare. Deck Builder Widget. Clive automation. Prepare. Deck Builder Widget. Clive automation. Deck Builder Widget. Deck Builder Widget. Clive automation. Deck Builder Widget. Deck Builde$ | ient method), 549   |
| method), 418   | $plot\_ternary\_cb()$ (AFL. automation. prepare. Sweep Builder Widget. Sweep Builder Widg     |
| pause() (AFL.automation.prepare.OT2Client.Client   | method), 73, 693, 711   |
| method), 482   | PneumaticPressureSampleCell (class in   |
| <pre>pause() (AFL.automation.prepare.OT2Client.OT2Client</pre>   | AFL. automation. loading. Pneumatic Pressure Sample Cell),  |
| method), 485   | 42, 284, 290  |
| pause() (AFL.automation.prepare.SampleSeriesWidget.Co  | li <b>Pn</b> veumaticSampleCell (class in   |
| method), 585   | AFL.automation.loading.PneumaticSampleCell),  |
| <pre>pause() (AFL.automation.sample.CastingServer.Client</pre>   | 44, 294, 300  |
| method), 724   | pop() (AFL.automation.APIServer.Driver.PersistentConfig   |
| pause() (AFL.automation.sample.CastingServer.OT2Client   | nt method), 171   |
| method), 729   | pop() (AFL.automation.APIServer.QueueDaemon.DataTrashcan  |
| <pre>pausePump() (AFL.automation.loading.ChemyxSyringePu</pre>   | ump.Chemy <b>n@thone</b> );til@p  |
| method), 36, 230, 233  | pop() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defaultdic   |
| permanent_session_lifetime   | method), 280  |
| (AFL.automation.APIServer.APIServer.Flask  | pop() (AFL.automation.loading.PneumaticPressureSampleCell.defaultdict   |
| attribute), 112  | method), 289  |
| PersistentConfig (class in   | pop() (AFL.automation.loading.PneumaticSampleCell.defaultdict   |
| AFL.automation.APIServer.Driver), 169  | method), 299  |
|  | pop() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdict  |
| AFL.automation.shared.PersistentConfig),   | method), 315  |
| 83, 773, 775   | pop() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict   |
| PipetteAction (class in AFL.automation.prepare), 386   | method), 326  |
|  | pop() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultdic   |
| AFL.automation.prepare.OT2Client), 486   | method), 372  |
|  | ephpk@l(AFL.automation.shared.DatasetWidget.defaultdict   |
| attribute), 443  | method), 756  |
|  | epop() (AFL.automation.shared.PersistentConfig.MutableMapping   |
| attribute), 461  | method), 772  |
|  | ulpop() (AFL.automation.shared.PersistentConfig.PersistentConfig  |
| attribute), 532  | method), 775  |
|  | expopitem() (AFL.automation.APIServer.Driver.PersistentConfig   |
| attribute), 555  | method), 171  |
|  | gptopikæm() (AFL.automation.APIServer.QueueDaemon.DataTrashcan  |
| attribute), 615  | method), 189  |
|  |   |

- popitem() (AFL.automation.loading.OneSelectorBlowoutSpostle@xHalrfa())di&FL.automation.loading.LoadStopperDriver.LoadStoppermethod), 280 method), 251
- popitem() (AFL.automation.loading.PneumaticPressureSappqsleCekledaftwel(dicAFL.automation.loading.OneSelectorBlowoutSampleCemethod), 290 method), 269
- popitem() (AFL.automation.loading.PneumaticSampleCelpdesfuuetdetaute() (AFL.automation.loading.OneSelectorBlowoutSampleCemethod), 300 method), 273
- popitem() (AFL.automation.loading.PushPullSelectorSamphsGelextefcuttel(at (AFL.automation.loading.OneSelectorBlowoutSampleCemethod), 316 method), 278
- popitem() (AFL.automation.loading.RSoXSSolutionSampleCell\_defaeduties() (AFL.automation.loading.PneumaticPressureSampleCell\_method), 327 method), 283
- popitem() (AFL.automation.loading.TwoSelectorBlowoutSposple@xHzdraQdiAtFL.automation.loading.PneumaticPressureSampleCelmethod), 373 method), 287
- popitem() (AFL.automation.shared.DatasetWidget.defaultapiost\_execute() (AFL.automation.loading.PneumaticSampleCell.Driver method), 756 method), 293
- popitem() (AFL.automation.shared.PersistentConfig.Muta**phrMapring**ute() (AFL.automation.loading.PneumaticSampleCell.PneumaticS
- popitem() (AFL.automation.shared.PersistentConfig.PersisperstConfig.Cute() (AFL.automation.loading.PushPullSelectorSampleCell.L method), 775 method), 308
- port (AFL.automation.APIServer.APIServer.ServiceInfo post\_execute() (AFL.automation.loading.PushPullSelectorSampleCell.Fattribute), 149 method), 312
- port (AFL.automation.shared.ServerDiscovery.AsyncServiceoute() (AFL.automation.loading.RSoXSSolutionSampleCell.Draattribute), 781 method), 319
- port (AFL.automation.shared.ServerDiscovery.ServiceInfo post\_execute() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoattribute), 799 method), 323
- post() (AFL.automation.APIServer.APIServer.Flask post\_execute() (AFL.automation.loading.TwoSelectorBlowoutSampleCemethod), 129 method), 365
- post\_execute() (AFL.automation.APIServer.DummyDrivpoDtivexecute() (AFL.automation.prepare.Dummy\_OT2\_Driver.Driver method), 175 method), 474
- post\_execute() (AFL:automation.APIServer.DummyDrivpoDunexpDuive() (AFL:automation.prepare.Dummy\_OT2\_Driver.Dummy\_method), 178 method), 478
  post\_execute() (AFL:automation.APIServer.DummyOT2Dviver.DummyO
- method), 182 method), 720
  post\_execute() (AFL.automation.APIServer.DummyOT2Dviver\_Execute()) (AFL.automation.sample.CastingServer.Driver
- post\_execute() (AFL.automation.APIServer.Dummy0T2**poster\_EnamyD()**WAFL.automation.sample.CastingServer.Driver method), 185 method), 726
- post\_execute() (AFL.automation.instrument.DummySASplositerexecute() (AFL.automation.sample\_env.TemperatureDeck.Driver method), 195 method), 733
- post\_execute() (AFL.automation.instrument.DummySASpartungsSelSute() (AFL.automation.sample\_env.TemperatureDeck.Tempera method), 735

  nost\_execute() (AFL automation instrument 122SAXS Driver, execute() (AFL automation APIServer Driver Driver)
- post\_execute() (AFL.automation.instrument.I22SAXS.Drpmae\_execute() (AFL.automation.APIServer.Driver.Driver.method), 201 method), 22, 168, 172
- post\_execute() (AFL.automation.instrument.SeabreezeUVVes.Dxecute() (AFL.automation.APIServer.DummyDriver.DummyDriver method), 207 method), 179
- post\_execute() (AFL.automation.instrument.SeabreezeU\f\f\executee(\f)\executee(\f)\executee(\f)\f\executee(\f)\
- method), 217

  post\_execute() (AFL.automation.instrument.SpecScreen\_pheiyexDriver() (AFL.automation.APIServer.DummyOT2Driver.DummyDtate), 185

  method), 185
- post\_execute() (AFL.automation.instrument.SpecScreen prejectSet) dua\_HIrinenomation.instrument.DummySAS.Driver method), 223 method), 195
- post\_execute() (AFL.automation.loading.LoadStopperDiprer\_Direct\_D

```
pre_execute() (AFL.automation.instrument.I22SAXS.Dripprepare_and_cast() (AFL.automation.sample.CastingServer.CastingSer
        method), 201
                                                              method), 89, 720, 730
pre_execute() (AFL.automation.instrument.I22SAXS.I22SAXSTare_casting_stocks()
                                                              (AFL.automation.sample.CastingServer.CastingServer
        method), 203
pre_execute() (AFL.automation.instrument.SeabreezeUVVis.Drivemethod), 89, 720, 730
        method), 207
                                                     PrepareWidget
                                                                                   (class
                                                                                                       in
pre_execute() (AFL.automation.instrument.SeabreezeUVVis.SeabreeELUWtimation.prepare.PrepareWidget),
                                                               69, 544, 565
        method), 217
pre_execute()(AFL.automation.instrument.SpecScreen_DriepaDeWiedget_Model
                                                                                       (class
                                                                                                       in
                                                              AFL.automation.prepare.PrepareWidget),
        method), 221
pre_execute() (AFL.automation.instrument.SpecScreen_Driver.Spe6Screen_Dbiver
                                                     PrepareWidget_View
        method), 223
                                                                                      (class
                                                                                                       in
pre_execute() (AFL.automation.loading.LoadStopperDriver.DriverAFL.automation.prepare.PrepareWidget),
        method), 248
                                                              69, 545, 565
pre_execute() (AFL.automation.loading.LoadStopperDripmelpRedStopperDiver
                                                                                                  module
                                                                                    (in
        method), 252
                                                              AFL.automation.prepare.PrepType), 488
pre_execute() (AFL.automation.loading.OneSelectorBlovponer$noqdes6elldQxiest()
                                                              (AFL.automation.APIServer.APIServer.Flask
        method), 269
pre_execute()(AFL.automation.loading.OneSelectorBlowoutSampleCell
                                                     PrepType (class in AFL.automation.prepare.Component),
        method), 273
pre_execute() (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell
        method), 278
                                                     PrepType (class in AFL.automation.prepare.PrepType),
pre_execute() (AFL.automation.loading.PneumaticPressureSampleCell_Driver
        method), 283
                                                     PressureController
                                                                                      (class
pre_execute() (AFL.automation.loading.PneumaticPressureSample\(\mathbb{Oell.AntenmationPhastdingSainttelOallPressureController\),
        method), 287
pre_execute() (AFL.automation.loading.PneumaticSampRacesbDreController
                                                                                      (class
                                                                                                       in
                                                              AFL.automation.loading.PressureController),
        method), 293
pre_execute() (AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCell
                                                     PressureController
        method), 297
                                                                                      (class
                                                                                                       in
pre_execute() (AFL.automation.loading.PushPullSelectorSampleCAlF.Daiwtomation.loading.UltimusVPressureController),
        method), 308
pre_execute() (AFL.automation.loading.PushPullSelectoPSwesplaCeADhPushPluESAksPtwwSampleCeUtlass
                                                              AFL.automation.loading.PressureControllerAsPump),
        method), 312
pre_execute() (AFL.automation.loading.RSoXSSolutionSampleCell4Driver, 305
        method), 319
                                                     prev_button_callback()
pre_execute() (AFL.automation.loading.RSoXSSolutionSampleCell,RFbXx88olutionSampleCell,ataLabelerWidget.DataLabelerWidget
        method), 323
                                                              method), 77, 740, 747
pre_execute() (AFL.automation.loading.TwoSelectorBlovpnerSabutta6alld2vlikback()
                                                              (AFL.automation.shared.DatasetWidget.DatasetWidget
        method), 365
pre_execute() (AFL.automation.loading.TwoSelectorBlowoutSample@thloTwoSelectorBlowoutSampleCell
        method), 370
                                                     prev_button_callback()
pre_execute() (AFL.automation.prepare.Dummy OT2 Driver.DriveAFL.automation.shared.DiffractionLabeler.DiffractionLabeler
        method), 474
                                                              method), 81, 760, 768
pre_execute() (AFL.automation.prepare.Dummy_OT2_DpinimeRimsny(WAF_Daintamation.loading.PneumaticPressureSampleCell.P
                                                              method), 44, 287, 291
        method), 478
pre_execute() (AFL.automation.sample.CastingServer.CaptingServer() (AFL.automation.loading.PneumaticSampleCell.Pneumatic
                                                              method), 45, 296, 301
        method), 720
pre_execute() (AFL.automation.sample.CastingServer.Dpreority (AFL.automation.APIServer.APIServer.ServiceInfo
                                                               attribute), 149
        method), 726
pre_execute() (AFL.automation.sample_env.Temperatureprixtant typeAFL.automation.shared.ServerDiscovery.AsyncServiceInfo
        method), 733
                                                               attribute), 781
pre_execute() (AFL.automation.sample_env.Temperature\(\textitetry)(ArtituralDevilation.shared.ServerDiscovery.ServiceInfo
        method), 735
                                                              attribute), 799
```

```
process_components()
                                                                                                                                                                       query_driver() (AFL.automation.loading.LoadStopperDriver.Client
                                                                                                                                                                                                    method), 246
                            (AFL.automation.prepare.MassBalance
                           method), 385
                                                                                                                                                                       query_driver() (AFL.automation.prepare.DeckBuilderWidget.Client
process_response() (AFL.automation.APIServer.APIServer.Flask method), 418
                           method), 132
                                                                                                                                                                        query_driver() (AFL.automation.prepare.OT2Client.Client
process_signal()(AFL.automation.loading.LoadStopperDriver.StopkthwodlQB\8\D
                                                                                                                                                                        query_driver() (AFL.automation.prepare.OT2Client.OT2Client
                           method), 257
process_signal()(AFL.automation.loading.LoadStopperDriver.StopEthnodIC,BIASS
                           method), 260
                                                                                                                                                                        query_driver() (AFL.automation.prepare.SampleSeriesWidget.Client
process_signal() (AFL.automation.loading.SensorCallbackThreadnsahsahsah)CathbackThread
                           method), 54, 343, 353
                                                                                                                                                                        query_driver() (AFL.automation.sample.CastingServer.Client
process_signal() (AFL.automation.loading.SensorCallbackThreadn&ihple\Tite\sholdCB
                                                                                                                                                                        query_driver() (AFL.automation.sample.CastingServer.OT2Client
                           method), 56, 345, 355
process_signal()(AFL.automation.loading.SensorCallbackThreadn&athpld)ad&Bv1
                           method), 55, 348, 354
                                                                                                                                                                        {\tt query\_scheduler} (AFL. automation. shared. Server Discovery. Async Service and the state of the state of
process_signal() (AFL.automation.loading.SensorCallbackThreadaStraipLite()dCBv2
                           method), 55, 351, 355
                                                                                                                                                                        query_scheduler(AFL.automation.shared.ServerDiscovery.ServiceBrown
product (class in AFL.automation.prepare.factory), 716
                                                                                                                                                                                                    attribute), 794
                                                                                                                                                                       queue_state() (AFL.automation.APIServer.APIServer.APIServer
propagate_exceptions
                                                                                                                                                                                                    method), 18, 100, 158
                           (AFL.automation.APIServer.APIServer.Flask
                           property), 115
                                                                                                                                                                        queue_state() (AFL.automation.APIServer.Client.Client
properties (AFL. automation. APIS erver. APIS erver. Service Info
                                                                                                                                                                                                   method), 20, 162, 165
                           attribute), 151
                                                                                                                                                                        queue_state() (AFL.automation.loading.LoadStopperDriver.Client
properties (AFL.automation.shared.ServerDiscovery.AsyncServiceImfethod), 246
                                                                                                                                                                        queue_state() (AFL.automation.prepare.DeckBuilderWidget.Client
                           attribute), 783
properties (AFL.automation.shared.ServerDiscovery.ServiceInfo method), 418
                           attribute), 801
                                                                                                                                                                        queue_state() (AFL.automation.prepare.OT2Client.Client
PROTECTED_SAMPLE_KEYS
                                                                                                                                                                                                   method), 482
                           (AFL.automation.APIServer.QueueDaemon.DataTyachequstate() (AFL.automation.prepare.OT2Client.OT2Client
                           attribute), 188
                                                                                                                                                                                                    method), 486
PROTECTED_SYSTEM_KEYS
                                                                                                                                                                        queue_state() (AFL.automation.prepare.SampleSeriesWidget.Client
                           (AFL.automation.APIServer.QueueDaemon.DataTrashcan method), 585
                           attribute), 188
                                                                                                                                                                        queue_state() (AFL.automation.sample.CastingServer.Client
{\tt pstr}\,(AFL. automation. prepare. Component. Parse Exception
                                                                                                                                                                                                    method), 724
                           attribute), 398
                                                                                                                                                                        queue_state() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                                     (class
PushPullSelectorSampleCell
                                                                                                                                                                                                   method), 730
                                                                                                                                                           in
                           AFL.automation.loading.PushPullSelectorSample@webled()
                                                                                                                                                                                                                  (AFL.automation.APIServer.Driver.Driver
                           47, 309, 316
                                                                                                                                                                                                    method), 21, 167, 171
                                  (AFL.automation.APIServer.APIServer.Flask queued() (AFL.automation.APIServer.DummyDriver.Driver
put()
                           method), 129
                                                                                                                                                                                                    method), 174
put() (AFL.automation.APIServer.APIServer.MutableQueuqueued() (AFL.automation.APIServer.DummyDriver.DummyDriver
                           method), 141
                                                                                                                                                                                                    method), 179
put () (AFL.automation.shared.MutableQueue.MutableQuequeued() (AFL.automation.APIServer.DummyOT2Driver.Driver
                           method), 82, 770, 771
                                                                                                                                                                                                    method), 181
                                                                                                                                                                        queued() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
Q
                                                                                                                                                                                                    method), 185
\verb|qsize()| (AFL. automation. APIS erver. APIS erver. Mutable Quantum ed () (AFL. automation. instrument. Dummy SAS. Driver and the property of the property 
                                                                                                                                                                                                    method), 194
                            method), 141
\verb|qsize()| (AFL. automation. shared. Mutable Queue. Mutable Queue()) (AFL. automation. instrument. Dummy SAS. Dummy SAS
                                                                                                                                                                                                     method), 197
                           method), 82, 770, 771
query_driver() (AFL.automation.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APIServer.APISe
                                                                                                                                                                                                    method), 200
                            method), 18, 99, 158
```

 $\verb"query_driver"() (AFL. automation. APIS erver. Client. Clien \texttt{A} \verb"ueued"() (AFL. automation. instrument. I22 SAXS. I22 SAXS) (AFL. automation. Instrument. I22 SAXS) (AFL. automation. I22 SAXS) (AFL. au$ 

method), 20, 161, 164

method), 203

```
queued() (AFL.automation.instrument.SeabreezeUVVis.Driver
                                                                                                                             25, 189, 192
                 method), 206
                                                                                                           queuehandler() (AFL.automation.EpicsADLiveProcess.AreaDetectorLive
queued() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVWishod), 26, 821
                                                                                                          quickbar() (AFL.automation.APIServer.Driver.Driver
                 method), 217
queued() (AFL.automation.instrument.SpecScreen_Driver.Driver
                                                                                                                            method), 21, 167, 171
                 method), 220
                                                                                                          quickbar() (AFL.automation.APIServer.DummyDriver.Driver
queued() (AFL.automation.instrument.SpecScreen_Driver.SpecScreemetholder, 174
                                                                                                           quickbar() (AFL.automation.APIServer.DummyDriver.DummyDriver
                 method), 223
queued() (AFL.automation.loading.LoadStopperDriver.Driver
                                                                                                                             method), 179
                                                                                                           quickbar() (AFL.automation.APIServer.DummyOT2Driver.Driver
                 method), 247
queued() (AFL.automation.loading.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriver.LoadStopperDriv
                                                                                                           quickbar() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
                 method), 252
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Durithard), 185
                                                                                                           quickbar() (AFL.automation.instrument.DummySAS.Driver
                 method), 268
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell. Onet/SedlectOrBlowoutSampleCell
                  method), 273
                                                                                                           quickbar() (AFL.automation.instrument.DummySAS.DummySAS
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell. TwebSedlectOrBlowoutSampleCell
                 method), 278
                                                                                                          quickbar() (AFL.automation.instrument.I22SAXS.Driver
queued() (AFL.automation.loading.PneumaticPressureSampleCell.Dniethod), 200
                 method), 282
                                                                                                           quickbar() (AFL.automation.instrument.I22SAXS.I22SAXS
queued() (AFL.automation.loading.PneumaticPressureSampleCell.PmeumaticPressureSampleCell
                 method), 287
                                                                                                           quickbar() (AFL.automation.instrument.SeabreezeUVVis.Driver
queued() (AFL.automation.loading.PneumaticSampleCell.Driver
                                                                                                                            method), 206
                                                                                                           quickbar() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVi
                 method), 292
queued() (AFL.automation.loading.PneumaticSampleCell.PneumatiaSathple)Cell7
                 method), 297
                                                                                                           quickbar() (AFL.automation.instrument.SpecScreen_Driver.Driver
queued() (AFL.automation.loading.PushPullSelectorSampleCell.Drimethod), 220
                                                                                                           quickbar() (AFL.automation.instrument.SpecScreen_Driver.SpecScreen_
                 method), 307
queued() (AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell
                 method), 312
                                                                                                           quickbar() (AFL.automation.loading.LoadStopperDriver.Driver
queued() (AFL.automation.loading.RSoXSSolutionSampleCell.Drivemethod), 248
                 method), 318
                                                                                                           quickbar() (AFL.automation.loading.LoadStopperDriver.LoadStopperDr
queued() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXfi&hadi)nSampleCell
                                                                                                           quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Dr.
                 method), 323
queued() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.Ducithcod), 268
                                                                                                          quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.On
                 method), 364
queued() (AFL.automation.loading.TwoSelectorBlowoutSampleCell. TweatSoil2ς(207BlowoutSampleCell
                 method), 371
                                                                                                           quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Two
queued() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
                                                                                                                            method), 278
                 method), 473
                                                                                                           quickbar() (AFL.automation.loading.PneumaticPressureSampleCell.Driv
queued() (AFL.automation.prepare.Dummy OT2 Driver.Dummy OT2etland)e282
                 method), 478
                                                                                                           {\tt quickbar()}\ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pneumatic Pne
{\tt queued()}\ (AFL. automation. sample. Casting Server. Casting Server
                                                                                                                            method), 287
                                                                                                           \verb"quickbar"()" (AFL. automation. loading. Pneumatic Sample Cell. Driver
                 method), 721
queued() (AFL. automation. sample. Casting Server. Driver
                                                                                                                             method), 292
                 method), 725
                                                                                                           quickbar() (AFL.automation.loading.PneumaticSampleCell.PneumaticSa
queued() (AFL.automation.sample_env.TemperatureDeck.Driver
                                                                                                                            method), 297
                                                                                                           quickbar() (AFL.automation.loading.PushPullSelectorSampleCell.Driver
                 method), 732
queued() (AFL.automation.sample_env.TemperatureDeck.TemperatumeDeckl), 307
                                                                                                           quickbar() (AFL.automation.loading.PushPullSelectorSampleCell.PushF
                 method), 735
                                                                                                                             method), 312
QueueDaemon
                                                         (class
                                                                                                  in
                 AFL.automation.APIServer.APIServer), 142
                                                                                                           quickbar() (AFL.automation.loading.RSoXSSolutionSampleCell.Driver
QueueDaemon
                                                         (class
                                                                                                  in
                                                                                                                             method), 318
```

AFL.automation.APIServer.QueueDaemon),

quickbar() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSS

read\_load\_buffer() (AFL.automation.loading.LoadStopperDriver.Sense

```
quickbar() (AFL.automation.loading.TwoSelectorBlowoutSampleCentalInively; 254
                                       method), 364
                                                                                                                                                                                                                                                    read_load_buffer() (AFL.automation.loading.SensorPollingThread.Sen
quickbar() (AFL.automation.loading.TwoSelectorBlowoutSampleCeMeRhowSelectorBlowotSampleCell
                                        method), 371
                                                                                                                                                                                                                                                     read_poll() (AFL.automation.loading.LoadStopperDriver.LoadStopperD
quickbar() (AFL.automation.prepare.Dummy OT2 Driver.Driver method), 39, 251, 262
                                                                                                                                                                                                                                                     read_poll_load() (AFL.automation.loading.LoadStopperDriver.LoadSto
                                       method), 473
quickbar() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_rotTa_odD;r3\0; 251, 262
                                        method), 478
                                                                                                                                                                                                                                                     read_sensor() (AFL.automation.loading.LoadStopperDriver.LoadStoppe
quickbar() (AFL.automation.sample.CastingServer.CastingServer method), 39, 251, 262
                                        method), 721
                                                                                                                                                                                                                                                     read_sensor() (AFL.automation.loading.PneumaticPressureSampleCell.
quickbar() (AFL.automation.sample.CastingServer.Driver
                                                                                                                                                                                                                                                                                             method), 44, 287, 291
                                       method), 725
                                                                                                                                                                                                                                                     read_sensor() (AFL.automation.loading.PneumaticSampleCell.Pneumat
quickbar() (AFL.automation.sample_env.TemperatureDeck.Driver_method), 45, 296, 301
                                                                                                                                                                                                                                                    read\_sensor\_poll() (AFL.automation.loading.PneumaticPressureSample)
                                        method), 732
quickbar() (AFL.automation.sample_env.TemperatureDeck.TemperatuethDeck, 288, 291
                                                                                                                                                                                                                                                    read_sensor_poll() (AFL.automation.loading.PneumaticSampleCell.Pn
                                       method), 735
quickbar_test() (AFL.automation.APIServer.DummyDriver.DummyDtivet), 45, 297, 301
                                        method), 23, 178, 180
                                                                                                                                                                                                                                                    read_sensor_poll_load()
quickbar_test() (AFL.automation.APIServer.DummyOT2Driver.D(MrhibyDutormation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.PneumaticPressureSampleCell.P
                                       method), 24, 184, 186
                                                                                                                                                                                                                                                                                             method), 44, 288, 291
quickbar_test2()(AFL.automation.APIServer.DummyDnieardDnensonDrivelt_load()
                                        method), 23, 178, 180
                                                                                                                                                                                                                                                                                             (AFL.automation.loading.PneumaticSampleCell.PneumaticSamp
quickbar_test2() (AFL.automation.APIServer.DummyOT2Driver.DummyDr45er297, 301
                                       method), 24, 184, 186
                                                                                                                                                                                                                                                    read_temp() (AFL.automation.sample_env.TemperatureDeck.Temperature
                                                                                                                                                                                                                                                                                             method), 75, 735, 736
R
                                                                                                                                                                                                                                                     read_text() (AFL.automation.instrument.SeabreezeUVVis.Path
ramp_dispense() (AFL.automation.loading.DigitalOutPressureConffell@DigitalOutPressureController
                                                                                                                                                                                                                                                     readlink() (AFL.automation.instrument.SeabreezeUVVis.Path
                                        method), 235
ramp_dispense() (AFL.automation.loading.DigitalOutPressureConffeller!PressureController
                                                                                                                                                                                                                                                     redirect() (AFL.automation.APIServer.APIServer.Flask
                                        method), 236
ramp_dispense() (AFL.automation.loading.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureCon
                                                                                                                                                                                                                                                     reduced() (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is
                                         method), 46, 302
ramp_dispense() (AFL.automation.loading.UltimusVPressureController
                                                                                                                                                                                                                                                    ReduceDaemon
                                                                                                                                                                                                                                                                                                                                                                                          (class
                                        method), 375
{\tt ramp\_dispense()} \ (AFL. automation. loading. Ultimus VP ressure Controller. {\tt QUARMASTOP} represented the property of t
                                                                                                                                                                                                                                                                                              27, 823, 826
                                         method), 377
read()(AFL.automation.loading.LoadStopperDriver.SensoFP9HstePurblueprint()
                                                                                                                                                                                                                                                                                             (AFL.automation.APIServer.APIServer.Flask
                                       method), 254
                                                                                                                                                                                                                                                                                             method), 119
read() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                                                    register_error_handler()
                                        method), 53, 334, 340
                                                                                                                                                                                                                                                                                              (AFL.automation.APIServer.APIServer.Flask
read() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                                                                                             method), 129
                                       method), 53, 337, 340
                                                                                                                                                                                                                                                    register\_service() (AFL.automation.APIServer.APIServer.Zeroconf
read()
                                                                      (AFL.automation.loading.Sensor.Sensor
                                                                                                                                                                                                                                                                                             method), 155
                                        method), 52, 339
\verb"read()" (AFL. automation. loading. Sensor Polling Thread. Sensor
                                                                                                                                                                                                                                                                                             method), 806
                                       method), 57, 357, 359
                                                                  (AFL. automation. prepare. Component DB \quad \texttt{relative\_to()} \ (AFL. automation. instrument. Seabreeze UVV is. Pathology of the property of th
read()
                                                                                                                                                                                                                                                                                              method), 214
                                       method), 382
\verb|read_bytes()| (AFL. automation. instrument. Seabreeze UVV \texttt{E-Parse}()| (AFL. automation. APIServer. APIServer. File Handler the state of the st
                                                                                                                                                                                                                                                                                              method), 105
                                         method), 211
{\tt read\_integrated()}\ (AFL. automation. instrument. I22SAX\$\ref{22348}\ref{22348})\ (AFL. automation. APIServer. APIServer. SMTPH and lertical contents of the state of the s
                                                                                                                                                                                                                                                                                              method), 147
                                        method), 29, 203, 204
```

*method*), 323

Index 901

reload() (AFL.automation.shared.widgetui.Markdown

| method), 818  | method), 636   |
|---|--|
| remove() (AFL.automation.APIServer.APIServer.Mutable Quemov   | e_class() (AFL.automation.prepare.SampleSeriesWidget.VBox        |
| method), 142  | method), 644   |
| remove() (AFL.automation.prepare.ComponentDB remove   | e_class() (AFL.automation.prepare.SweepBuilderWidget.Button      |
| method), 382  | method), 657   |
| remove() (AFL.automation.shared.MutableQueue  | e_class() (AFL.automation.prepare.SweepBuilderWidget.Checkb      |
| method), 82, 770, 771   | method), 665   |
|   | e_class() (AFL.automation.prepare.SweepBuilderWidget.HBox        |
| (AFL.automation.APIServer.APIServer.Zeroconf  | method), 673   |
| · · · · · · · · · · · · · · · · · · ·   | e_class() (AFL.automation.prepare.SweepBuilderWidget.Label       |
| remove_all_service_listeners()  | method), 681   |
|   | e_class() (AFL.automation.prepare.SweepBuilderWidget.Text        |
| method), 806  | method), 701   |
| remove_class() (AFL.automation.prepare.DeckBuilderWirlemDB  |  |
| method), 406  | method), 709   |
|   |  |
| remove_class() (AFL.automation.prepare.DeckBuilderWirlgmb@  |  |
| method), 414  | method), 19, 100, 158  |
| remove_class() (AFL.automation.prepare.DeckBuilderWirdgmoD  | •  |
| method), 427  | method), 20, 162, 165  |
| remove_class() (AFL.automation.prepare.DeckBuilderWirdenbHd   |  |
| method), 435  | method), 246   |
| ${\tt remove\_class()} \ (AFL. automation. prepare. Deck Builder \textit{Wirdeymodu} and \textit{prepare} and \textit{prepare} are also becomes a substantial property of the prop$                                   | 1 1  |
| method), 443  | method), 418   |
| remove_class() (AFL.automation.prepare.DeckBuilderWindgmoVe   | * *  |
| method), 461  | method), 482   |
| remove_class() (AFL.automation.prepare.DeckBuilderWindgmoVe   | Boxtem() (AFL.automation.prepare.OT2Client.OT2Client             |
| method), 469  | method), 486   |
| remove_class() (AFL.automation.prepare.PrepareWidgetzPentroom   | e_item() (AFL.automation.prepare.SampleSeriesWidget.Client       |
| method), 498  | method), 585   |
| remove_class() (AFL.automation.prepare.PrepareWidgetn@hook  | boitem() (AFL.automation.sample.CastingServer.Client             |
| method), 506  | method), 724   |
| remove_class() (AFL.automation.prepare.PrepareWidgetremove  | by intem() (AFL. automation. sample. Casting Server. OT 2 Client |
| method), 516  | method), 730   |
| remove_class() (AFL.automation.prepare.PrepareWidgetHeRox   | · · · · · · · · · · · · · · · · · · ·                            |
| method), 524  | method), 19, 100, 158  |
| remove_class() (AFL.automation.prepare.PrepareWidgetxlenlock)   |  |
| method), 532  | method), 156   |
| remove_class() (AFL.automation.prepare.PrepareWidgetTemove  |  |
| method), 555  | method), 807   |
| remove_class() (AFL.automation.prepare.PrepareWidgetr\\(\text{effave}\)   |  |
|   |  |
| method), 563  | method), 787   |
| remove_class() (AFL.automation.prepare.SampleSeriesWidgerM  |  |
| method), 572  | (AFL.automation.APIServer.APIServer.Zeroconf                     |
| remove_class() (AFL.automation.prepare.SampleSeriesWidget.C   |  |
|   | e_service_listener()   |
| remove_class() (AFL.automation.prepare.SampleSeriesWidget.F   |  |
| method), 591  | method), 806   |
| ${\tt remove\_class()} \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Sample Series {\it Wild great Minimum Series}) \ (AFL. automation. prepare. Prepare. Prepare. Prepare. Prepare. Prepare. Prepare. Prepare. Prepare. $ |  |
| method), 599  | method), 548   |
| ${\tt remove\_class()}\ (AFL. automation. prepare. Sample Series {\it Windge and Minimum M$   |  |
| method), 607  | method), 72, 648, 650  |
| ${\tt remove\_class()}\ (AFL. automation. prepare. Sample Series {\it Widge of Matter}) \\$   | edbædrtical_lines()  |
| method), 615  | (AFL. automation. shared. Data Labeler Widget. Data Labeler View |
| remove_class() (AFL.automation.prepare.SampleSeriesWidget.T   | Textmethod), 77, 739, 748  |

```
remove_vertical_lines()
                                                                                                                                                                                                                                                                      method), 39, 251, 262
                                      (AFL.automation.shared.DiffractionLabeler.DiffraceiserLabretepYtenrgets()
                                    method), 81, 762, 769
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Dr
Removed (AFL.automation.shared.ServerDiscovery.ServiceStateChangeethod), 66, 476, 478
                                     attribute), 802
                                                                                                                                                                                                                                reset_pump() (AFL.automation.loading.PneumaticSampleCell.Pneumatic
removeFilter() (AFL.automation.APIServer.APIServer.FileHandlemethod), 45, 296, 300
                                                                                                                                                                                                                                reset_queue_daemon()
                                    method), 105
removeFilter() (AFL.automation.APIServer.APIServer.SMTPHandlAFL.automation.APIServer.APIServer.APIServer
                                     method), 147
                                                                                                                                                                                                                                                                      method), 18, 99, 157
rename()(AFL.automation.instrument.SeabreezeUVVis.Patteset_queue_daemon()
                                    method), 212
                                                                                                                                                                                                                                                                      (AFL.automation.APIServer.Client.Client
rename_component()(AFL.automation.prepare.factory.Solution
                                                                                                                                                                                                                                                                     method), 20, 161, 164
                                    method), 715
                                                                                                                                                                                                                                 reset_queue_daemon()
rename_component()(AFL.automation.prepare.Solution
                                                                                                                                                                                                                                                                      (AFL.automation.loading.LoadStopperDriver.Client
                                     method), 391
                                                                                                                                                                                                                                                                       method), 246
render_template()
                                                                                                                                  (in
                                                                                                                                                                                           module reset_queue_daemon()
                                    AFL.automation.APIServer.APIServer), 94
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.DeckBuilderWidget.Client
render_unqueued() (AFL.automation.APIServer.APIServer.APIServeethod), 418
                                    method), 18, 100, 158
                                                                                                                                                                                                                                reset_queue_daemon()
reorder_queue() (AFL.automation.APIServer.APIServer.APIServer(AFL.automation.prepare.OT2Client.Client
                                    method), 19, 100, 158
                                                                                                                                                                                                                                                                      method), 482
replace() (AFL.automation.instrument.SeabreezeUVVis.Pndset_queue_daemon()
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.OT2Client.OT2Client
                                     method), 212
request() (AFL.automation.APIServer.APIServer.ServiceInfo
                                                                                                                                                                                                                                                                      method), 486
                                                                                                                                                                                                                                 reset_queue_daemon()
                                    method), 151
request() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfAFL.automation.prepare.SampleSeriesWidget.Client
                                                                                                                                                                                                                                                                      method), 584
                                     method), 783
request() (AFL.automation.shared.ServerDiscovery.Servirels@t_queue_daemon()
                                    method), 801
                                                                                                                                                                                                                                                                      (AFL.automation.sample.CastingServer.Client
request_class(AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                      method), 723
                                     attribute), 111
                                                                                                                                                                                                                                 reset_queue_daemon()
{\tt request\_context()} \ (AFL. automation. APIS erver. APIS erver. Flask \ \ (AFL. automation. sample. Casting Server. OT 2 Client 1 and 
                                     method), 133
                                                                                                                                                                                                                                                                      method), 730
reset() (AFL.automation.APIServer.QueueDaemon.DataTresetusample() (AFL.automation.APIServer.Driver.Driver
                                     method), 189
                                                                                                                                                                                                                                                                      method), 22, 168, 172
reset() (AFL. automation. EpicsADLiveProcess. Client. Clieneset_sample() (AFL. automation. APIServer. DummyDriver. Driver.
                                    method), 27, 822, 823
                                                                                                                                                                                                                                                                      method), 175
reset() (AFL.automation.loading.LoadStopperDriver.LoadStopperBampde() (AFL.automation.APIServer.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyDriver.DummyD
                                    method), 39, 251, 262
                                                                                                                                                                                                                                                                      method), 179
                                                                   (AFL.automation.prepare.MassBalance reset_sample() (AFL.automation.APIServer.DummyOT2Driver.Driver
reset()
                                    method), 385
                                                                                                                                                                                                                                                                     method), 182
                                                                  (AFL. automation. prepare. Sample Series reset\_sample() (AFL. automation. API Server. Dummy OT2 Driver. Dummy III) and the sample of the sam
reset()
                                    method), 388
                                                                                                                                                                                                                                                                      method), 185
reset_dataset() (AFL.automation.shared.DatasetWidgetdestetsetStindgete() (AFL.automation.APIServer.QueueDaemon.DataTrashca
                                     method), 79, 751, 757
                                                                                                                                                                                                                                                                      method), 189
reset_dataset() (AFL.automation.shared.DatasetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDasterset
                                     method), 79, 752, 758
                                                                                                                                                                                                                                                                      method), 195
reset_load_buffer()
                                                                                                                                                                                                                                reset_sample() (AFL.automation.instrument.DummySAS.DummySAS
                                      (AFL.automation.loading.LoadStopperDriver.SensorPollingThethad), 197
                                                                                                                                                                                                                                \verb"reset_sample()" (AFL. automation. instrument. I22SAXS. Driver)" and the sample () is a simple of the sample of
                                    method), 254
reset_load_buffer()
                                                                                                                                                                                                                                                                      method), 201
                                    (AFL.automation.loading.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.Se
                                    method), 57, 357, 359
                                                                                                                                                                                                                                                                     method), 204
reset_pol1() (AFL.automation.loading.LoadStopperDrivarelsardStampdel())i(AFL.automation.instrument.SeabreezeUVVis.Driver
```

*method*), 385

| J. D. 207   | J. D. 20, 252, 262   |
|---|--|
| method), 207 reset_sample() (AFL.automation.instrument.SeabreezeUV&se&eath  | method), 39, 252, 262  |
| method), 217  | (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSe   |
| reset_sample() (AFL.automation.instrument.SpecScreen_Driver.D   | •  |
|   | ank_levels()   |
| reset_sample() (AFL.automation.instrument.SpecScreen_Driver.Sp  |  |
|   | <b>xxx.bu.r.com<u>o</u>mnuvon</b> .todatng.OnesetectorBlowoutsampleCett.1wose<br>method), 277  |
|   |  |
| reset_sample() (AFL.automation.loading.LoadStopperDniesenD  |  |
| method), 248  | (AFL:automation.loading.PneumaticPressureSampleCell.Pneuma   |
| reset_sample() (AFL.automation.loading.LoadStopperDriver.Load   |  |
|   | ank_levels()   |
| reset_sample() (AFL.automation.loading.OneSelectorBlowoutSam  | ž'.  |
|   | method), 45, 296, 300  |
| reset_sample() (AFL.automation.loading.OneSelectorBlanesserfatta  |  |
| method), 273  | (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolut   |
| $\verb"reset_sample()" (AFL. automation. loading. One Selector Blowout Sample()") and the selection of the sele$  |  |
|   | ank_levels()   |
| $\verb"reset_sample()" (AFL. automation. loading. Pneumatic Pressure Sample()") and the sample of the s$  |  |
|   | method), 59, 369, 374  |
| $\verb"reset_sample()" (AFL. automation. loading. Pneumatic Presentation of the presentat$  | <b>dr GetlsKi)</b> eumati <b>(AFds.suneSoonipln Getl</b> pare.Deck   |
| method), 287  | method), 384   |
| reset_sample() (AFL.automation.loading.PneumaticSamplesCedL_D   | arigets() (AFL.automation.prepare.MassBalance  |
| method), 293  | method), 385   |
| reset_sample() (AFL.automation.loading.PneumaticSamplesedLP.  | <mark>นางเกเรา โดย (ApfelCall</mark> tomation.prepare.Dummy_OT2_Driver.Dumn  |
|   | method), 66, 477, 479  |
| reset_sample() (AFL.automation.loading.PushPullSelectveScentpte   |  |
|   | method), 67, 485, 487  |
| reset_sample()(AFL.automation.loading.PushPullSelectveScentple  |  |
|   | method), 729   |
| reset_sample() (AFL.automation.loading.RSoXSSolutionsample()  |  |
|   | method), 547   |
| reset_sample() (AFL.automation.loading.RSoXSSolution <b>Sample()</b>  |  |
|   | method), 70, 629, 646  |
| reset_sample() (AFL.automation.loading.TwoSelectorBlaveraSum  |  |
|   | method), 211   |
| reset_sample() (AFL.automation.loading.TwoSelectorBlavesapSns   |  |
|   | attribute), 111  |
| reset_sample() (AFL.automation.prepare.Dummy_OT2_Destarb)   |  |
| •   | method), 36, 230, 234  |
|   |  |
| reset_sample() (AFL.automation.prepare.Dummy_OT2_DebæieDu   | • =  |
|   | method), 19, 100, 158  |
| reset_sample() (AFL.automation.sample.CastingServer.CoestingServer  | = · · · · ·  |
|   | method), 21, 162, 165  |
| reset_sample() (AFL.automation.sample.CastingServer.Dnetwoiev   | - · · · · ·  |
|   | method), 22, 168, 172  |
| reset_sample() (AFL.automation.sample_env.TemperatureDexkeD   | the state of the s |
|   | method), 175   |
| $\verb"reset_sample()" (AFL. automation. sample\_env. Temperature \textit{Pex.} ke \textit{Temperature} \textit{Temperature} \textit{Pex.} ke Temper$ |  |
|   | method), 179   |
| reset_stocks() (AFL.automation.prepare.Deck retrieve  | e_obj()(AFL.automation.APIServer.DummyOT2Driver.Driver   |
| method), 384  | method), 182   |
| ${\tt reset\_stocks()}\ (AFL. automation. prepare. Mass Balance\ {\tt retrieve}$  | e_obj()(AFL.automation.APIServer.DummyOT2Driver.DummyI   |

904 Index

 ${\tt reset\_stopper()} \ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. loading. Load Stopper {\tt Dreitwilewed Stopper}) \\ (AFL. automation. load Stopper {\tt Dreitwilewed Stopper}) \\$ 

method), 185

| <i>method</i> ), 195 | method), 482  |
|----------------------|---|
| retrieve_obj()(AFL   | .automation.instrument.DummySAS. <b>rPetminySeA_S</b> obj() (AFL.automation.prepare.OT2Client.OT2Client   |
| method), 197         | method), 486  |
| retrieve_obj()(AFL   | .automation.instrument.I22SAXS.Drietrieve_obj() (AFL.automation.prepare.SampleSeriesWidget.Client   |
| method), 201         | method), 585  |
| retrieve_obj()(AFL   | .automation.instrument.I22SAXS.I2 <b>2%AXS</b> eve_obj() (AFL.automation.sample.CastingServer.CastingServer   |
| method), 204         | method), 721  |
| retrieve_obj()(AFL   | .automation.instrument.SeabreezeUY&txDeinerobj() (AFL.automation.sample.CastingServer.Client  |
| method), 207         | method), 724  |
| retrieve_obj()(AFL   | .automation.instrument.SeabreezeU <b>Y&amp;ixSexlor_cobcjUYVAF</b> L.automation.sample.CastingServer.Driver   |
| method), 217         | method), 726  |
| retrieve_obj()(AFL   | .automation.instrument.SpecScreen_tletiviteDeivebj() (AFL.automation.sample.CastingServer.OT2Client   |
| method), 221         | method), 730  |
| retrieve_obj()(AFL   | .automation.instrument.SpecScreen_tetiviespecStricon_Abilivantomation.sample_env.TemperatureDeck.Driver   |
| method), 223         | method), 733  |
| retrieve_obj()(AFL   | automation.loading.LoadStopperDniearGlivet_obj()(AFL.automation.sample_env.TemperatureDeck.Tempera.   |
| method), 246         | method), 735  |
|                      | automation.loading.LoadStopperDnieeneDtiver(AFL.automation.APIServer.Driver.PersistentConfig  |
| method), 248         | method), 171  |
|                      | .automation.loading.LoadStopperDniveneExtadStAFflenDniveartion.shared.PersistentConfig.PersistentConfig   |
| method), 252         | method), 84, 775, 776   |
|                      | .automation.loading.OneSelectorBla <b>vernalSad</b> up <b>toKell_Drixe</b> ler()  |
| method), 269         | (AFL.automation.APIServer.APIServer.JWTManager  |
| **                   | .automation.loading.OneSelectorBlowoutSam <b>ple(i)eM.Qh∂S</b> electorBlowoutSampleCell   |
| method), 273         | RFC   |
| **                   | .automation.loading.OneSelectorBlowou <b>RXO</b> np <b>22E</b> &lPTwoSelectorBlowoutSampleCell  |
| method), 278         | rglob() (AFL.automation.instrument.SeabreezeUVVis.Path  |
| **                   | .automation.loading.PneumaticPressureSampl <b>n@bbD</b> r@el  |
| method), 283         | right (AFL.automation.prepare.DeckBuilderWidget.Layout  |
| **                   | .automation.loading.PneumaticPressureSampl <b>aGebluPa)e</b> AAAaticPressureSampleCell  |
| method), 287         | right (AFL.automation.prepare.PrepareWidget.Layout  |
| **                   | .automation.loading.PneumaticSampleCell.Dicitieribute), 538   |
| method), 293         | right (AFL.automation.prepare.SampleSeriesWidget.Layout   |
| **                   | .automation.loading.PneumaticSampleCell.Pn <b>etwibatieS</b> ampleCell  |
| method), 297         | right (AFL.automation.prepare.SweepBuilderWidget.Layout   |
|                      | automation.loading.PushPullSelectorSampleGuhiDuts)e;1687  |
| method), 308         | rinseAll() (AFL.automation.loading.OneSelectorBlowoutSampleCell.On  |
| **                   | .automation.loading.PushPullSelectorSample <b>GeddhBd3hPu3</b> lSelectorSampleCell  |
| method), 312         | rinseAll() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Tw  |
| **                   | .automation.loading.RSoXSSolutionSampleCethIdthiody; 277  |
| method), 319         | rinseAll() (AFL automation . loading . Pneumatic Pressure Sample Cell . Pneumatic Pressure Sample Pressure Sample Pressure Sample |
| **                   | automation.loading.RSoXSSolutionSampleCeh <b>iaRSoX</b> SSolutionSampleCell   |
| method), 323         | rinseAll() (AFL automation . loading . Pneumatic Sample Cell . Pneumatic Sample Sample Cell . Pneumatic Sample |
|                      | automation.loading.TwoSelectorBlowoutSam <b>pleDellD</b> ASe296, 300  |
| method), 365         | rinseAll() (AFL.automation.loading.PushPullSelectorSampleCell.PushP   |
|                      | .automation.loading.TwoSelectorBlowoutSam <b>pleDell.Tw8SelectorBl</b> owoutSampleCell  |
|                      | rinseAll() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSo.   |
| method), 371         | automation.prepare.DeckBuilderWidget.Clientiethod), 50, 323, 328.   |
| • " '                |   |
| method), 419         | rinseAll() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.Two   |
|                      | automation.prepare.Dummy_OT2_Driver.Driverthod), 60, 370, 374   |
| method), 474         | rinseCatch() (AFL.automation.loading.OneSelectorBlowoutSampleCell.  |
|                      | automation.prepare.Dummy_OT2_Driver.DummthoOJL2_Driver  |
| <i>method</i> ), 478 | rinseCatch() (AFL.automation.loading.OneSelectorBlowoutSampleCell.'   |

method), 277

 $\verb"retrieve_obj()" (AFL. automation. prepare. OT 2 Client. Client$ 

```
rinseCatch() (AFL.automation.loading.PushPullSelectors@oople@ath.PASHP.automSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelec
                            method), 48, 313, 317
                                                                                                                                                                                                       attribute), 130
rinseCatch() (AFL.automation.loading.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RSoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolutionSarpolet@ED.RsoXSolution
                            method), 50, 323, 328
                                                                                                                                                                                                       method), 129
rinseCatch() (AFL.automation.loading.TwoSelectorBlow&SS&SpddCttlich&Ss&pdaCeBlowoutSampleCssll
                           method), 60, 370, 374
                                                                                                                                                                                                      AFL.automation.loading.RSoXSSolutionSampleCell),
rinseCell() (AFL.automation.loading.OneSelectorBlowoutSampleCell. DNeSelectorBlowoutSampleCell
                                                                                                                                                                          run() (AFL.automation.APIServer.APIServer.APIServer
                           method), 273
rinseCell() (AFL.automation.loading.OneSelectorBlowoutSampleGnldtflod)Sell&cldtBlodosellocotorBlowoutSampleCell
                                                                                                                                                                                                             (AFL.automation.APIServer.APIServer.Flask
                           method), 277
                                                                                                                                                                          run()
rinseCell() (AFL.automation.loading.PneumaticPressureSampleColleHund)maticPressureSampleCell
                            method), 44, 286, 291
                                                                                                                                                                          run() (AFL.automation.APIServer.APIServer.QueueDaemon
rinseCell() (AFL.automation.loading.PneumaticSampleCell.PneumatibSampleCell
                           method), 45, 296, 300
                                                                                                                                                                          run() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
rinseCell() (AFL.automation.loading.PushPullSelectorSampleCelltRetsidRi)(ISE)&ACSampleCell
                            method), 48, 312, 317
                                                                                                                                                                          run() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaem
rinseCell() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXISSolutionSampleCell
                           method), 50, 323, 328
                                                                                                                                                                          run() (AFL.automation.loading.LoadStopperDriver.SensorPollingThread
rinseCell() (AFL.automation.loading.TwoSelectorBlowoutSampleGulttlind)SelEctorBlowoutSampleCell
                           method), 60, 370, 374
                                                                                                                                                                          run() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
rinseCellFlood() (AFL.automation.loading.OneSelectorBlowoutSampheQell2OneSelectorBlowoutSampleCell
                           method), 273
                                                                                                                                                                          run() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
rinseCellFlood() (AFL.automation.loading.OneSelectorBlowoutSampheQellTwoSelectorBlowoutSampleCell
                                                                                                                                                                          run() (AFL.automation.loading.Sensor.DummySensor1
                            method), 277
rinseCellFlood() (AFL.automation.loading.PushPullSelectorSampleOeblPuthPublSeteCtorSampleCell
                           method), 48, 312, 317
                                                                                                                                                                          run() (AFL.automation.loading.Sensor.DummySensor2
rinseCellFlood() (AFL.automation.loading.RSoXSSolutionSampleGielhBSpXSS\dbBitionSumpleCell
                           method), 50, 323, 328
                                                                                                                                                                          {\tt run()}\ (AFL. automation. loading. Sensor Callback Thread. Sensor Callbac
rinseCellFlood() (AFL.automation.loading.TwoSelectorBlowoutSampleQell5fwoSelectorBlowoutSampleCell
                           method), 60, 370, 374
                                                                                                                                                                          run() (AFL.automation.loading.SensorCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThresholdCallbackThread.SimpleThreshold
rinseCellPull() (AFL.automation.loading.OneSelectorBlowoutSampltfOell),OM6SelectorBlowoutSampleCell
                            method), 273
                                                                                                                                                                          run() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
rinseCellPull() (AFL.automation.loading.OneSelectorBlowoutSamplttGell, TstribSelectorBlowoutSampleCell
                                                                                                                                                                          run() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
                            method), 277
rinseCellPull() (AFL.automation.loading.PushPullSelectorSampleCellOHiyshPullSelectorSampleCell
                           method), 48, 312, 317
                                                                                                                                                                          run() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
rinseCellPull() (AFL.automation.loading.RSoXSSolutionSampleQulttRSd(XSSolb@ionSSampleCell
                            method), 50, 323, 328
                                                                                                                                                                          run() (AFL.automation.shared.DataLabelerWidget.DataLabelerView
rinseCellPull() (AFL.automation.loading.TwoSelectorBlowoutSampltfGell,TN8qSelectOrBlowoutSampleCell
                           method), 60, 370, 374
                                                                                                                                                                          run() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
rinseSyringe() (AFL.automation.loading.OneSelectorBlowoutSampleCiell.QndSdletpOrtBlowoutSampleCell
                            method), 273
                                                                                                                                                                          run() (AFL.automation.shared.DatasetWidget.DatasetWidget
rinseSyringe() (AFL.automation.loading.OneSelectorBlowoutSampleCell\)W\\SelEc\tilde{O}\)EllowoutSampleCell
                                                                                                                                                                          run() (AFL.automation.shared.DatasetWidget.DatasetWidget_View
                           method), 277
rinseSyringe() (AFL.automation.loading.PushPullSelectorSample@edlhDd\hB\hJ\S\equiv \text{Et\delta}SampleCell}
                            method), 48, 312, 317
                                                                                                                                                                          run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabeler
rinseSyringe() (AFL.automation.loading.RSoXSSolutionSampleCethARSocX)$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{Solution}$\sqrt{So
                           method), 50, 322, 328
                                                                                                                                                                          run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView
rinseSyringe() (AFL.automation.loading.TwoSelectorBlowoutSampleChell J.WaSellectorBlowoutSampleCell
                            method), 60, 370, 374
                                                                                                                                                                          run() (AFL.automation.shared.ServerDiscovery.RunThread
rmdir() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                       method), 85, 790, 808
                                                                                                                                                                          run() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                           method), 212
{\tt root} (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                       method), 796
                           property), 214
                                                                                                                                                                          run_threaded() (AFL.automation.APIServer.APIServer.APIServer
```

| method), 18, 99, 157 RunThread (class in AFL.automation.shared.ServerDiscove   | save_cb() (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidgery), method), 65, 419, 471   |
|--|--|
| 84, 788, 808   | save_cb() (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget method), 509   |
| S  | save_cb() (AFL.automation.prepare.PrepareWidget.StockBuilderWidget   |
| sa_aio_discover_server_by_name()   | method), 548   |
| (AFL.automation.APIServer.Client.ServerDiscove   | epave_cb() (AFL.automation.prepare.StockBuilderWidget.StockBuilderWid  |
| class method), 163   | method), 72, 648, 650  |
| sa_aio_discover_server_by_name()   | SeabreezeUVVis (class in   |
| (AFL.automation.shared.ServerDiscovery.Servers class method), 85, 792, 809   | Discovery AFL.automation.instrument.SeabreezeUVVis), 30, 215, 218  |
| sa_discover_server_by_name()   | secret_key (AFL.automation.APIServer.APIServer.Flask   |
| (AFL. automation. API Server. Client. Server Discover and the property of th     |  |
| class method), 163   | select_jinja_autoescape()  |
| sa_discover_server_by_name()   | (AFL.automation.APIServer.APIServer.Flask  |
| (AFL. automation. shared. Server Discovery. Server Discovery Server Discover Discovery Server Discovery Server Discovery Server Discovery Server Discover Discov     | Discovery method), 116   |
| class method), 85, 792, 809  | $\verb selectPort( )  (AFL. automation. loading. Cetoni MultiPos Valve. Cetoni MultiPos Valve.$         |
| SainSmartRelay (class in   | method), 34, 226, 227  |
| AFL.automation.loading.SainSmartRelay), 51, 330, 331   | <pre>selectPort() (AFL.automation.loading.CetoniMultiPosValve.FlowSelect<br/>method), 227</pre>  |
| samefile()(AFL.automation.instrument.SeabreezeUVVis  | $\textbf{\textit{PallectPort()}} \ (AFL. automation. loading. Double Vici Multipos Selector. Double Vici$ |
| method), 211   | method), 37, 238, 240  |
| Sample (class in AFL.automation.prepare), 387  | $\verb selectPort()  (AFL. automation. loading. Double Vici Multipos Selector. Flow that the property of the prop$         |
| sample_composition_space()   | method), 238   |
| (AFL.automation.prepare.MassBalance  | $\verb selectPort()  (AFL. automation. loading. Double Vici Multipos Selector. Vi$         |
| method), 386   | method), 239   |
| SampleCell (class in AFL.automation.loading.PneumaticF   | PESSAFET SPORT 160 (AF, L. automation. loading. Flow Selector. Flow Selector method), 38, 243  |
|  | A = A + A + A + A + A + A + A + A + A +  |
| 298  | method), 378   |
| ${\tt SampleCell}\ (class\ in\ AFL. automation. loading. PushPullSet and the property of the pr$ | ISA STARPET (MAFL. automation. loading. ViciMultipos Selector. ViciMultipo   |
| 313  | method), 61, 379, 380  |
| SampleCell (class in AFL.automation.loading.RSoXSSolut 324   | method), 157   |
| SampleCell (class in AFL.automation.loading.SampleCell) 52, 333  | )send() (AFL.automation.prepare.DeckBuilderWidget.Button method), 406  |
| SampleCell (class in AFL.automation.loading.TwoSelector  | r <b>Band (in SAFil-Jacony</b> ation.prepare.DeckBuilderWidget.Checkbox  |
| 366  | method), 414   |
| SampleSeries (class in AFL.automation.prepare), 388  | send() (AFL.automation.prepare.DeckBuilderWidget.Dropdown  |
| SampleSeriesTool_reset_cb()  | method), 427   |
| (AFL.automation.prepare.PrepareWidget.Prepare  | e Nang(4) (AFL.automation.prepare.DeckBuilderWidget.HBox   |
| method), 69, 544, 565  | method), 435   |
| SampleSeriesWidget (class in   | send() (AFL.automation.prepare.DeckBuilderWidget.Label method), 443  |
| AFL.automation.prepare.PrepareWidget),   | send() (AFL.automation.prepare.DeckBuilderWidget.Layout  |
| 545  | method), 453   |
| SampleSeriesWidget (class in   | send() (AFL.automation.prepare.DeckBuilderWidget.Text  |
| AFL.automation.prepare.SampleSeriesWidget), 70, 627, 646   | method), 461   |
| SampleSeriesWidget_Model (class in   | send() (AFL.automation.prepare.DeckBuilderWidget.VBox  |
| AFL. automation. prepare. Sample Series Widget),   | method), 469   |
| 71, 629, 646   | send() (AFL.automation.prepare.PrepareWidget.Button  |
| SampleSeriesWidget_View (class in  | method), 498   |
| AFL.automation.prepare.SampleSeriesWidget), 71, 629, 646   | send() (AFL.automation.prepare.PrepareWidget.Checkbox method), 506   |
| , ,  |  |

 $\verb"send()" (AFL. automation. prepare. Prepare Widget. Dropdown") \\$ 

| method), 516                                 | ${\tt send\_deck\_cb()}\ (AFL. automation. prepare. Prepare Widget. Deck Builder automation. Prepare Widget. Prepare Widget. Deck Builder automation. Prepare Widget. Prepare Wi$ |
|--|--|
| send() (AFL.automation.prepare.PrepareWidg   | et.HBox method), 509   |
| method), 524                                 | <pre>send_deck_config() (AFL.automation.prepare.Deck</pre>   |
| send() (AFL.automation.prepare.PrepareWidg   |  |
| method), 532                                 | ${\sf send\_deck\_config()}\ (AFL. automation. prepare. Deck Builder Widget. Deck Builder Widget$ |
| send() (AFL.automation.prepare.PrepareWidge  |  |
| method), 542                                 | send_file() (in module   |
| send() (AFL.automation.prepare.PrepareWid    |  |
| method), 555                                 | send_file_max_age_default  |
| send() (AFL.automation.prepare.PrepareWidg   |  |
| method), 563                                 | property), 112   |
| send() (AFL.automation.prepare.SampleSeriesW |  |
| method), 572                                 | method), 383   |
|  | (idget.Checkband_state() (AFL.automation.prepare.DeckBuilderWidget.Button  |
| method), 580                                 | method), 406   |
|  | (idget.Floas Find_state() (AFL.automation.prepare.DeckBuilderWidget.Checkbox   |
| method), 591                                 | method), 414   |
|  | Vidget.HBossend_state() (AFL.automation.prepare.DeckBuilderWidget.Dropdown   |
| method), 599                                 | method), 427   |
|  | Vidget.IntTestend_state() (AFL.automation.prepare.DeckBuilderWidget.HBox   |
| method), 607                                 | method), 435   |
|  | Vidget.Labedend_state() (AFL.automation.prepare.DeckBuilderWidget.Label  |
| method), 615                                 | method), 443<br>/idget.Layasænd_state() (AFL.automation.prepare.DeckBuilderWidget.Layout   |
| method), 625                                 | method), 453   |
|  | idget.Text send_state() (AFL.automation.prepare.DeckBuilderWidget.Text   |
| method), 636                                 | method), 461   |
|  | (idget.VBosend_state() (AFL.automation.prepare.DeckBuilderWidget.VBox  |
| method), 644                                 | method), 469   |
|  | Vidget.Buttoend_state() (AFL.automation.prepare.PrepareWidget.Button   |
| method), 657                                 | method), 498   |
|  | Vidget.Che <b>skind</b> :_state()(AFL.automation.prepare.PrepareWidget.Checkbox  |
| method), 665                                 | method), 506   |
| send() (AFL.automation.prepare.SweepBuilderV | Vidget.HBoxend_state() (AFL.automation.prepare.PrepareWidget.Dropdown  |
| method), 673                                 | method), 516   |
| send() (AFL.automation.prepare.SweepBuilderV | Vidget.Labelend_state()(AFL.automation.prepare.PrepareWidget.HBox  |
| method), 681                                 | method), 524   |
| send() (AFL.automation.prepare.SweepBuilderV | Vidget.Lay&mend_state()(AFL.automation.prepare.PrepareWidget.Label   |
| method), 691                                 | method), 532   |
|  | Vidget.Textsend_state() (AFL.automation.prepare.PrepareWidget.Layout   |
| method), 701                                 | method), 542   |
|  | Vidget.VBosend_state() (AFL.automation.prepare.PrepareWidget.Text  |
| method), 709                                 | method), 555   |
| •  | v.Zeroconfsend_state() (AFL.automation.prepare.PrepareWidget.VBox  |
| method), 807                                 | method), 563   |
|  | PIServer. A Besidry state() (AFL. automation. prepare. Sample Series Widget. Button  |
| method), 18, 100, 158                        | method), 572   |
| send_array_as_jpg()                          | send_state() (AFL.automation.prepare.SampleSeriesWidget.Checkbox   |
| (AFL.automation.APIServer.APIServer.         |  |
| method), 18, 100, 158                        | send_state() (AFL.automation.prepare.SampleSeriesWidget.FloatText  |
|  | nusVPressureController   |
| method), 60, 376, 377                        | send_state() (AFL.automation.prepare.SampleSeriesWidget.HBox   |
| send_deck_cb() (AFL.automation.prepare.Deci  | Dunaer waget. Deck <b>inanaer j</b> y <b>i</b> aiget   |

method), 65, 419, 471

| <pre>send_state() (AFL.automation.prepare.SampleSeriesW</pre>  | /idget.IntText54, 341, 353   |   |   |
|--|--|---|---|
| method), 607   | SensorPollingThread  | (class  | in  |
| $send\_state() (AFL. automation. prepare. Sample Series Water () (AFL. automation. prepare. Sample Series ()$    | idget.Label AFL.automation.lo  | oading.LoadStoppe   | rDriver),   |
| method), 615   | 252  |   |   |
| $send\_state() (AFL. automation. prepare. Sample Series Water () (AFL. automation. prepare. Sample Series ()$    | /id <b>§en/soxRo</b> llingThread   | (class  | in  |
| method), 625   | AFL. $automation.lo$   | oading.SensorPollii   | ıgThread),  |
| $send\_state()$ (AFL.automation.prepare.SampleSeriesW  | Vidget.Text 56, 356, 358   |   |   |
| method), 636   | SerialCommsException, $8$  | 6, 360, 810   |   |
| $send\_state()$ (AFL.automation.prepare.SampleSeriesW  |  | (class  | in  |
| method), 644   | AFL. $automation.lo$   | pading.DummyPum   | pp),  |
| $send\_state()$ (AFL.automation.prepare.SweepBuilderV  | Vidget.Button241   |   |   |
| method), 657   | SerialDevice   | (class  | in  |
| <pre>send_state() (AFL.automation.prepare.SweepBuilderV</pre>  | Vidget.Check <b>!A&amp;</b> L.automation.lo  | pading.NE1kSyring   | ePump),   |
| method), 665   | 265  |   |   |
| send_state()(AFL.automation.prepare.SweepBuilderV  | _  | (class  | in  |
| method), 673   | AFL.automation.la  | oading.PressureCo   | ntrollerAsPump),  |
| <pre>send_state() (AFL.automation.prepare.SweepBuilderV</pre>  |  |   |   |
| method), 681   | SerialDevice   | (class  | in  |
| $\verb send_state()  (AFL. automation. prepare. SweepBuilder Variable SweepBuilder Sw$ | ${\it Vidget. Layou}$ ${\it AFL. automation. late}$  | oading.SainSmartR   | elay),  |
| method), 691   | 331  |   |   |
| send_state()(AFL.automation.prepare.SweepBuilderV  |  | (class  | in  |
| method), 701   | AFL. $automation.lo$   | oading.SerialDevic  | e), 57,   |
| send_state()(AFL.automation.prepare.SweepBuilderV  |  |   |   |
| method), 709   | SerialDevice   | (class  | in  |
| $\verb send_static_file()  (AFL. automation. APIS erver. APIS)  $  | Server.Flask AFL.automation.la   | oading.ViciMultipo  | sSelector),   |
| method), 129   | 378  |   |   |
| $\verb sendCommand()  (AFL. automation. loading. Chemyx Syring) $  |  | (in   | module  |
| method), 36, 229, 233  | AFL. $automation.sh$   |   | ), 86,  |
| $\verb sendCommand()  (AFL. automation. loading. Double Vici Material Command ())  (AFL. automati$ | •  |   |   |
| method), 239   | (4.77  | DIG A DIG   |   |
|  | server (AFL.automation.AF  | AServer.APIServer   | :ServiceInfo  |
| $\verb sendCommand()  (AFL. automation. loading. Dummy Pump) $   | SerialDevic <b>e</b> attribute), 149   |   | ·   |
| sendCommand() (AFL.automation.loading.DummyPump method), 242   | SerialDevic <b>e</b> ttribute), 149<br>server (AFL.automation.sh   |   | ·   |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel  | SerialDevicættribute), 149<br>server (AFL.automation.sh<br>Pump.SerialDætnibute), 781  | ared.ServerDiscove  | ery.AsyncServiceInfo  |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265  | SerialDevic <b>e</b> ttribute), 149<br>server (AFL.automation.sh<br>Pump.Serial <b>Dev</b> iikute), 781<br>server (AFL.automation.sh   | ared.ServerDiscove  | ery.AsyncServiceInfo  |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265<br>sendCommand() (AFL.automation.loading.PressureCont  | SerialDevicettribute), 149 server (AFL.automation.sh. Pump.SerialDeviibute), 781 server (AFL.automation.sh. rollerAsPumpt&ibiadDevi&   | ared.ServerDiscove<br>ared.ServerDiscove  | ery.AsyncServiceInfo  |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265<br>sendCommand() (AFL.automation.loading.PressureCont<br>method), 304  | SerialDevicettribute), 149 server (AFL.automation.sh. Pump.SerialDetnibute), 781 server (AFL.automation.sh. rollerAsPumptBibibiadDevice server_cmd() (AFL.automa   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl  | ery.AsyncServiceInfo  |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265<br>sendCommand() (AFL.automation.loading.PressureCont<br>method), 304<br>sendCommand() (AFL.automation.loading.SainSmartRel  | SerialDevicettribute), 149 server (AFL.automation.she Pump.SerialDetnibute), 781 server (AFL.automation.she rollerAsPumptSeibiadd)eVice server_cmd() (AFL.automat ay.SainSmartiRatayd), 20, 161,   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331   | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDevidute), 781 server (AFL.automation.sh. rollerAsPumptbebiaddDevide server_cmd() (AFL.automation.sh. ay.SainSmartbebod), 20, 161, server_cmd() (AFL.automation.sh.   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client  |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265<br>sendCommand() (AFL.automation.loading.PressureCont<br>method), 304<br>sendCommand() (AFL.automation.loading.SainSmartRel  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDeviidaute), 781 server (AFL.automation.sh. rollerAsPumptBribiadDevites server_cmd() (AFL.automation.sh. ay.SainSmartRedugd), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDeviverthod), 246  | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client   |
| sendCommand() (AFL.automation.loading.DummyPump<br>method), 242<br>sendCommand() (AFL.automation.loading.NE1kSyringel<br>method), 265<br>sendCommand() (AFL.automation.loading.PressureCont<br>method), 304<br>sendCommand() (AFL.automation.loading.SainSmartRel<br>method), 331<br>sendCommand() (AFL.automation.loading.SainSmartRel<br>method), 331  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDeviibute), 781 server (AFL.automation.sh. rollerAsPumptSiibiadDevite server_cmd() (AFL.automa ay.SainSmartiRalayd), 20, 161, server_cmd() (AFL.automa ay.SerialDevivethod), 246 server_cmd() (AFL.automa   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client   |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.  | SerialDevicentribute), 149 server (AFL.automation.she Pump.SerialDetniibute), 781 server (AFL.automation.she rollerAsPumptSribiand)evi20e server_cmd() (AFL.automa ay.SainSmartiRethood), 20, 161, server_cmd() (AFL.automa ay.SerialDevivethod), 246 server_cmd() (AFL.automa SerialDevicemethod), 418  | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360  | SerialDevicentribute), 149 server (AFL.automation.she Pump.SerialDetnibute), 781 server (AFL.automation.she rollerAsPumptSiabiand)evisse server_cmd() (AFL.automation.she ay.SainSmartisahood), 20, 161, server_cmd() (AFL.automation.she ay.SerialDevicenthod), 246 server_cmd() (AFL.automation.she) SerialDevicemethod), 418 server_cmd() (AFL.automation.she)  | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.  | SerialDevicentribute), 149 server (AFL.automation.she Pump.SerialDetribute), 781 server (AFL.automation.she rollerAsPumptSirbiadDevite server_cmd() (AFL.automation.she ay.SainSmartRethood), 20, 161, server_cmd() (AFL.automation.she ay.SerialDevicenthod), 246 server_cmd() (AFL.automation.she SerialDevicemethod), 418 server_cmd() (AFL.automation.she SerialDevicemethod), 418 server_cmd() (AFL.automation.she)   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load<br>ation.prepare.Deck   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378  | SerialDevicentribute), 149 server (AFL.automation.she Pump.SerialDeviidaute), 781 server (AFL.automation.she rollerAsPumptBribiadDeviidae), 202 server_cmd() (AFL.automation.she ay.SainSmartRedayd), 20, 161, server_cmd() (AFL.automation.she ay.SerialDevincethod), 246 server_cmd() (AFL.automation.she SerialDevicemethod), 418 server_cmd() (AFL.automation.she Selector.SerialDevivology, 482 server_cmd() (AFL.automation.she)   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load<br>ation.prepare.Deck   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378  | SerialDevicentribute), 149 server (AFL. automation. she Pump. SerialDeviidaute), 781 server (AFL. automation. she roller As Pumpt BribinalDeviidae server_cmd() (AFL. automation. she ay. Sain Smarthedluyd), 20, 161, server_cmd() (AFL. automation. she ay. SerialDevicethod), 246 server_cmd() (AFL. automation. she SerialDevicemethod), 418 server_cmd() (AFL. automation. she Selector. SerialDevice), 482 server_cmd() (AFL. automation. she Selector. Vici Medilipal Selector  | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load<br>ation.prepare.Deck<br>ation.prepare.OT20   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDeviibute), 781 server (AFL.automation.sh. rollerAsPumptSiibiadDevi&& server_cmd() (AFL.automation.sh. ay.SainSmartiRalayd), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDeviocthod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.sh. Selector.SerialDeviocthod), 482 server_cmd() (AFL.automation.sh. Selector.ViciMultipalS&&&otor server_cmd() (AFL.automation.sh.   | ared.ServerDiscove<br>ared.ServerDiscove<br>ation.APIServer.Cl<br>164<br>ation.loading.Load<br>ation.prepare.Deck<br>ation.prepare.OT20   | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDetniibute), 781 server (AFL.automation.sh. rollerAsPumptSribiand)evi20e server_cmd() (AFL.automation.sh. ay.SainSmartiRathayd), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDevivethod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.sh. Selector.SerialDetnion.sh. Selector.VicilMuttlipalsSelector server_cmd() (AFL.automation.sh. Selector.VicilMuttlipalsSelector server_cmd() (AFL.automation.sh.)   | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.OT20  | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.Client  |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelimethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelimethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelimethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSimethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSimethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDetniibute), 781 server (AFL.automation.sh. rollerAsPumptSribiald)evise server_cmd() (AFL.automation.sh. ay.SainSmartRethood), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDevivethod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.sh. Selector.SerialDetniod), 482 server_cmd() (AFL.automation.sh. Selector.VicinnethipalsSelector server_cmd() (AFL.automation.sh. server_cmd() (AFL.automation.sh.), 584 server_cmd() (AFL.automation.sh.  | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp  | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client                                    |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339 sensor_reset() (AFL.automation.loading.PneumaticP.  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDetniibute), 781 server (AFL.automation.sh. rollerAsPumptSiabiand)evites server_cmd() (AFL.automation.sh. ay.SainSmartikatuod), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDevincethod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.sh. Selector.SerialDethod), 482 server_cmd() (AFL.automation.server_cmd() (AFL.automaticn.server_cmd() (AFL.automation.server_cmd() (AFL.automaticn.server_cmd() | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp ation.sample.Castin cressureSample.Castin                        | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client                                    |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339 sensor_reset() (AFL.automation.loading.PneumaticPmethod), 44, 288, 291  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDeviidaute), 781 server (AFL.automation.sh. rollerAsPumptBibiadDeviidae server_cmd() (AFL.automation.sh. ay.SainSmartRadayd), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDevicethod), 246 server_cmd() (AFL.automation.sh. SerialDevicethod), 448 server_cmd() (AFL.automation.sh. Selector.SerialDevicethod), 482 server_cmd() (AFL.automation.sh. Selector.ViciMediapatSelector server_cmd() (AFL.automation.sh.   | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp ation.sample.Castin bressureSample.Castin ation.sample.Castin    | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client                                    |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339 sensor_reset() (AFL.automation.loading.PneumaticPmethod), 44, 288, 291 sensor_reset() (AFL.automation.loading.PneumaticSensor).   | SerialDevicentribute), 149 server (AFL.automation.she Pump.SerialDeviidaute), 781 server (AFL.automation.she rollerAsPumpt&ibiadDevi@e server_cmd() (AFL.automa ay.SainSmartkalund), 20, 161, server_cmd() (AFL.automa ay.SerialDevicethod), 246 server_cmd() (AFL.automa SerialDevicemethod), 418 server_cmd() (AFL.automa SerialDevicemethod), 482 server_cmd() (AFL.automa Selector.SerialDevicethod), 482 server_cmd() (AFL.automa Selector.ViciMedipod), \$600 server_cmd() (AFL.automa method), 584 server_cmd() (AFL.automa ressureSampla@bbbapp@maticP server_cmd() (AFL.automa method), 584   | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp ation.sample.Castic tressureSample.Castic                        | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client ingServer.Client                   |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339 sensor_reset() (AFL.automation.loading.PneumaticPmethod), 44, 288, 291 sensor_reset() (AFL.automation.loading.PneumaticSemethod), 45, 297, 301  | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDeviibute), 781 server (AFL.automation.sh. rollerAsPumptSibiandDevite server_cmd() (AFL.automation.sh. ay.SainSmartiRalayd), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDeviocthod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.sh. Selector.SerialDeviocthod), 482 server_cmd() (AFL.automation.sh. Selector.ViciMultipal)Selector server_cmd() (AFL.automation.sh. server_key (AFL.automation.sh.   | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp ation.sample.Castic tressureSample.Castic                        | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client ingServer.Client                   |
| sendCommand() (AFL.automation.loading.DummyPumpmethod), 242 sendCommand() (AFL.automation.loading.NE1kSyringelmethod), 265 sendCommand() (AFL.automation.loading.PressureContmethod), 304 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SainSmartRelmethod), 331 sendCommand() (AFL.automation.loading.SerialDevice.method), 57, 360 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 378 sendCommand() (AFL.automation.loading.ViciMultiposSmethod), 380 Sensor (class in AFL.automation.loading.Sensor), 52, 339 sensor_reset() (AFL.automation.loading.PneumaticPmethod), 44, 288, 291 sensor_reset() (AFL.automation.loading.PneumaticSensor).   | SerialDevicentribute), 149 server (AFL.automation.sh. Pump.SerialDetniibute), 781 server (AFL.automation.sh. rollerAsPumptSiibiand)evise server_cmd() (AFL.automation.sh. ay.SainSmartiRation), 20, 161, server_cmd() (AFL.automation.sh. ay.SerialDeviocethod), 246 server_cmd() (AFL.automation.sh. SerialDevicemethod), 418 server_cmd() (AFL.automation.server_cmd() (AFL.automation.server_key (AFL.automation.se | ared.ServerDiscoverared.ServerDiscoveration.APIServer.Cl 164 ation.loading.Load ation.prepare.Deck ation.prepare.OT20 ation.prepare.Samp ation.sample.Castion cressureSample.Castion ll on.APIServer.APIS | ery.AsyncServiceInfo ery.ServiceInfo ient.Client StopperDriver.Client BuilderWidget.Client Client.Client Client.OT2Client pleSeriesWidget.Client ingServer.Client eryer.ServiceInfo |

| att              | ribute), 781                                |                               |               | method), 204  |              |
|------------------|---|-------------------------------|---------------|---|--------------|
| server_key       | (AFL.automation.sh                          | ared.ServerDiscover           | y.Ser         | erv <b>setInfonf</b> ig()(AFL.automation.instrument.SeabreezeUVVis.Driver   |              |
| att              | ribute), 799                                |                               |               | method), 207  |              |
| ServerDisc       | covery                                      | (class                        | in            | ${\tt set\_config()}\ (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is$   | UV           |
| AF               | FL.automation.APISe                         | rver.Client), 162             |               | method), 217  |              |
| ServerDisc       | covery                                      | (class                        | in            | ${\tt set\_config()}\ (AFL. automation. instrument. Spec Screen\_Driver. Driver)$   |              |
|                  | FL. $automation$ . $share a$                | d.ServerDiscovery),           |               | method), 220  |              |
| 85               | , 791, 808                                  |                               |               | $set\_config() (AFL. automation. instrument. Spec Screen\_Driver. S$ | ee           |
|                  | tate_changed                                |                               |               | method), 223  |              |
| (A.              | FL.automation.share                         | d.ServerDiscovery.A           | syncS         | cS&+tcathmfisg() (AFL.automation.loading.LoadStopperDriver.Client   |              |
|                  | ribute), 779                                |                               |               | method), 246  |              |
|                  | tate_changed                                |                               |               | ${\tt set\_config()}\ (AFL. automation. loading. Load Stopper Driver. Driver$   |              |
|                  |   | d.ServerDiscovery.Se          | ervice        | ceBrowser method), 248  |              |
|                  | ribute), 796                                |                               |               | $set\_config() (AFL. automation. loading. Load Stopper Driver. Load Stop$   | er           |
| ServiceBro       |   | (class                        | in            | · · · · · · · · · · · · · · · · · · ·   |              |
| AF               | FL.automation.sharea                        | d.ServerDiscovery),           |               | $\verb set_config()  (AFL. automation. loading. One Selector Blowout Sample Cells and Selector Blowout Sample C$  | ll.          |
| 79               |   |                               |               | method), 269  |              |
| ServiceInf<br>AF | Fo (d<br>FL.automation.APISe                | class<br>rver.APIServer), 147 |               | <pre>set_config() (AFL.automation.loading.OneSelectorBlowoutSampleCel<br/>method), 274</pre>  | ll.          |
| ServiceInf       | Fo (a                                       | class                         | in            | ${\tt set\_config()}\ (AFL. automation. loading. One Selector Blowout Sample Cells and the setting of the setting$   | ll.          |
| AF               | FL.automation.sharea                        | d.ServerDiscovery),           |               | method), 278  |              |
| 79               | 7   |                               |               | ${\tt set\_config()}\ (AFL. automation. loading. Pneumatic Pressure Sample Cell$  | L            |
| ServiceSta       | ateChange                                   | (class                        | in            | method), 282  |              |
| AF<br>80         | FL.automation.sharea<br>2.                  | d.ServerDiscovery),           |               | <pre>set_config() (AFL.automation.loading.PneumaticPressureSampleCell     method), 287</pre>  | .P           |
| session_co       |   |                               |               | set_config() (AFL.automation.loading.PneumaticSampleCell.Driver   |              |
|                  | FL.automation.APIS                          | erver.APIServer.Flas          | k             | method), 293  |              |
|                  | operty), 112                                |                               |               | set_config()(AFL.automation.loading.PneumaticSampleCell.Pneuma  | ıtic         |
| _                |   | mation.APIServer.AF           | IServ         | rver.Flask method), 297   |              |
|                  | ribute), 113                                |                               |               | <pre>set_config() (AFL.automation.loading.PushPullSelectorSampleCell.D</pre>  | )ri          |
| set_access       | s_cookies()                                 | (in mod                       | lule          | method), 307  |              |
| AF               | FL.automation.APISe                         | rver.APIServer), 95           |               | ${\tt set\_config()}\ (AFL. automation. loading. PushPullSelector Sample Cell. Particle (Configuration of the Configuration) and the configuration of the Configuration (Configuration) and the Configuration (Configuration) an$   | us           |
| set_aspira       | ate_rate()                                  |                               |               | method), 312  |              |
|                  | FL.automation.prepa<br>ethod), 67, 477, 479 | are.Dummy_OT2_Dr              | iver.L        | DsattnyconF1.gQpi(AFL.automation.loading.RSoXSSolutionSampleCell.Drimethod), 319  | ve           |
|                  |   | .APIServer.Client.Cl          | ient          | set_config() (AFL.automation.loading.RSoXSSolutionSampleCell.RSo  | οX           |
|                  | ethod), 20, 161, 164                        |                               |               | method), 324  |              |
|                  |   | .APIServer.Driver.Dr          | river         | r set_config() (AFL.automation.loading.TwoSelectorBlowoutSampleCel  | ll.I         |
|                  | ethod), 21, 167, 171                        |                               |               | method), 364  |              |
|                  |   | .APIServer.DummvD             | river.        | er. <b>Det</b> ve <b>c</b> onfig() (AFL.automation.loading.TwoSelectorBlowoutSampleCel  | <i>ll</i> .' |
| _                | ethod), 174                                 |                               |               | method), 371  |              |
|                  | * *   | .APIServer.DummvD             | river.        | er. <b>Datua phfing()</b> (AFL. automation. prepare. DeckBuilder Widget. Client   |              |
|                  | ethod), 179                                 | Ý                             |               | method), 418  |              |
|                  |   | .APIServer.DummyC             | T2Di          | Driet Dointig() (AFL.automation.prepare.Dummy_OT2_Driver.Driver   |              |
| me               | ethod), 182                                 | •                             |               | method), 473  |              |
| set_config       | () (AFL.automation                          | .APIServer.DummyC             | T2Di          | Dri <b>set_DronshyD(r)</b> v@FL.automation.prepare.Dummy_0T2_Driver.Dummy_  | 0            |
|                  | ethod), 185                                 | •                             |               | method), 478  |              |
| set_config       | g() (AFL.automation                         | .instrument.DummyS            | AS.D          | Drsetr_config() (AFL.automation.prepare.OT2Client.Client  |              |
| me               | ethod), 194                                 |                               |               | method), 482  |              |
| set_config       | g() (AFL.automation                         | .instrument.DummyS            | AS.D          | Dispersive Schooling () (AFL. automation. prepare. OT2Client. OT2Client   |              |
| me               | ethod), 198                                 | ·                             |               | method), 486  |              |
| set_config       | g() (AFL.automation                         | instrument.I22SAXS            | .Driv         | iveset_config() (AFL.automation.prepare.SampleSeriesWidget.Client   |              |
| me               | ethod), 200                                 |                               |               | method), 584  |              |
| set_config       | g() (AFL.automation                         | instrument.I22SAXS            | . <i>I22S</i> | 2SASES_config() (AFL.automation.sample.CastingServer.CastingServer  |              |

| method), 721  | method), 297   |
|---|--|
| set_config() (AFL.automation.sample.CastingServer.Clieset_dat   |  |
| method), 723  | method), 308   |
| <pre>set_config() (AFL.automation.sample.CastingServer.Driset_dat</pre>   |  |
| method), 725  | method), 313   |
| set_config() (AFL.automation.sample.CastingServer.OT <b>26tiedt</b> at  |  |
| method), 730  | method), 319   |
| set_config() (AFL.automation.sample_env.TemperatureDack_Idart   |  |
| method), 732  | method), 324   |
| set_config() (AFL.automation.sample_env.TemperatureDack_Than,   | method), 365   |
| <pre>method), 736 set_data() (AFL.automation.APIServer.Driver set_dat</pre>   |  |
| method), 22, 168, 172   | method), 371   |
| set_data() (AFL.automation.APIServer.DummyDriver.Driset_dat   |  |
| method), 175  | method), 474   |
| set_data()(AFL.automation.APIServer.DummyDriver.DusanyDan   |  |
| method), 179  | method), 478   |
| set_data()(AFL.automation.APIServer.DummyOT2Driveseridat  |  |
| method), 182  | method), 721   |
| set_data()(AFL.automation.APIServer.DummyOT2Driveseurdamy   |  |
| method), 185  | method), 726   |
| set_data()(AFL.automation.instrument.DummySAS.Driv&et_dat   |  |
| method), 195  | method), 733   |
| set_data()(AFL.automation.instrument.DummySAS.DumseySAGat   |  |
| method), 198  | method), 736   |
| set_data()(AFL.automation.instrument.I22SAXS.Driver set_dec   |  |
| method), 201  | method), 65, 419, 471  |
| set_data()(AFL.automation.instrument.I22SAXS.I22SAXSet_dec  |  |
| method), 204  | method), 509   |
| set_data()(AFL.automation.instrument.SeabreezeUVVis.Derbedis  |  |
| method), 207  | (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Dr                      |
| set_data()(AFL.automation.instrument.SeabreezeUVVis.Seabreez  |  |
|   | ver_object()   |
| set_data()(AFL.automation.instrument.SpecScreen_Driver.Driver   |  |
| method), 221  | method), 19, 100, 158  |
| set_data()(AFL.automation.instrument.SpecScreen_DrivseSpecSize  |  |
| method), 223  | (AFL.automation.APIServer.Client.Client                                    |
| set_data() (AFL.automation.loading.LoadStopperDriver.Driver   | method), 20, 162, 165  |
|   | ver_object()   |
| <pre>set_data() (AFL.automation.loading.LoadStopperDriver.LoadStop</pre>  |  |
| method), 252  | method), 246   |
| set_data() (AFL.automation.loading.OneSelectorBlowoutSetupdro   |  |
| method), 269  | (AFL.automation.prepare.DeckBuilderWidget.Client                           |
| <pre>set_data() (AFL.automation.loading.OneSelectorBlowoutSampleC</pre>   |  |
| method), 274 set_dri  | ver_object()   |
| $\mathtt{set\_data()}\ (AFL. automation. loading. One Selector Blowout Sample Control of the Control o$ | Cell <b>A IF I</b> va <b>Stelenctwiti Bhopræpt Sær QT2Cell</b> ent. Client |
| method), 278  | method), 482   |
| set_data()(AFL.automation.loading.PneumaticPressureSweetpletCi  | elleld <u>r</u> iobiject()   |
| method), 283  | (AFL.automation.prepare.OT2Client.OT2Client                                |
| $\verb set_data()  (AFL. automation. loading. Pneumatic Pressure Sample Center Continuous C$    | el <b>ln&amp;tleod</b> ha#i&BressureSampleCell                             |
| method), 287 set_dri  | ver_object()   |
| $\verb set_data()  (AFL. automation. loading. Pneumatic Sample Cell. Driven and the property of t$    | · (AFL.automation.prepare.SampleSeriesWidget.Client                        |
| method), 293  | method), 585   |
| set_data()(AFL.automation.loading.PneumaticSampleCesletPneumaticSampleCe        | meir Sobijele CEN  |

*method*), 246

method), 248

| (AFL.automation.sample.CastingServer.Client method), 252   |
|--|
| method), 724 set_object() (AFL.automation.loading.OneSelectorBlowoutSampleCell.  |
| set_driver_object() method), 269   |
| (AFL.automation.sample.CastingServer.OT2Clienset_object() (AFL.automation.loading.OneSelectorBlowoutSampleCell.  |
| method), 730 method), 274  |
| set_gantry_speed() (AFL.automation.prepare.Dummy_GE2_DwjectDijn(AiF_LOTI20_iDartiven.loading.OneSelectorBlowoutSampleCell.   |
| method), 67, 477, 479  method), 278  |
| set_mass() (AFL.automation.prepare.Component.Component_object() (AFL.automation.loading.PneumaticPressureSampleCell.L  |
| method), 64, 396, 398 method), 283 set_mass() (AFL.automation.prepare.factory.Solution set_object() (AFL.automation.loading.PneumaticPressureSampleCell.F  |
| method), 715 method), 288  |
| set_mass() (AFL.automation.prepare.Solute method), set_object() (AFL.automation.loading.PneumaticSampleCell.Driver   |
| 390 method), 293   |
| set_mass() (AFL.automation.prepare.Solution set_object()(AFL.automation.loading.PneumaticSampleCell.Pneumati   |
| method), 391 method), 297  |
| set_mass() (AFL.automation.prepare.Solvent method), set_object() (AFL.automation.loading.PushPullSelectorSampleCell.Dru  |
| 394 <i>method</i> ), 308   |
| set_name() (AFL.automation.APIServer.APIServer.FileHandlerobject() (AFL.automation.loading.PushPullSelectorSampleCell.PushPushPullSelectorSampleCell.PushPushPullSelectorSampleCell.Pus   |
| method), 105 method), 313  |
| set_name() (AFL.automation.APIServer.APIServer.SMTPIsendlerbject() (AFL.automation.loading.RSoXSSolutionSampleCell.Drive   |
| method), 147 method), 319  |
| $\verb set_object()  (AFL. automation. APIS erver. Client. Client   \verb set_object()  (AFL. automation. loading. RSoXS Solution Sample Cell. RSoXIII)   Continued to the set_object() (AFL. automation. loading. RSoXS Solution Sample Cell. RSoXIIII)   Continued to the set_object() (AFL. automation. loading. RSoXS Solution Sample Cell. RSoXIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII$  |
| method), 21, 162, 165 method), 324   |
| $\verb set_object()  (AFL. automation. APIS erver. Driver. Bet_object()) (AFL. automation. loading. Two Selector Blowout Sample Cell. Bet_object()) (AFL. automation. APIS erver. Driver. Driver. Driver. Bet_object()) (AFL. automation. APIS erver. Driver. Dri$   |
| method), 22, 168, 172 method), 365   |
| set_object() (AFL.automation.APIServer.DummyDriver. <b>Det</b> tyeobject() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.   |
| method), 371   |
| set_object() (AFL.automation.APIServer.DummyDriver.BethuohDeixet() (AFL.automation.prepare.DeckBuilderWidget.Client  |
| method), 179  method), 419  OTA Discussion (CAFI), and the control of the control   |
| set_object() (AFL.automation.APIServer.DummyOT2Driset_Dibjact() (AFL.automation.prepare.Dummy_OT2_Driver.Driver  |
| method), 182 method), 474 set_object() (AFL.automation.APIServer.DummyOT2Driset_Dibject())vAFL.automation.prepare.Dummy_OT2_Driver.Dummy_O   |
| method), 185 method), 478  |
| set_object() (AFL.automation.instrument.DummySAS.Drsetr_object() (AFL.automation.prepare.OT2Client.Client  |
| method), 195  method), 482   |
| set_object() (AFL.automation.instrument.DummySAS.DusentySAS) Description (AFL.automation.prepare.OT2Client.OT2Client   |
| method), 198 method), 486  |
| set_object() (AFL.automation.instrument.122SAXS.Driveret_object() (AFL.automation.prepare.SampleSeriesWidget.Client  |
| method), 201 method), 585  |
| set_object() (AFL.automation.instrument.I22SAXS.I22SAXS_object() (AFL.automation.sample.CastingServer.CastingServer  |
| method), 204 method), 721  |
| set_object() (AFL.automation.instrument.SeabreezeUVVseDrobrject() (AFL.automation.sample.CastingServer.Client  |
| method), 207 method), 724  |
| set_object() (AFL.automation.instrument.SeabreezeUVV\$:e\$eabreezeUV |
| method), 218 method), 726  |
| set_object() (AFL.automation.instrument.SpecScreen_Dsect() (AFL.automation.sample.CastingServer.OT2Client  |
| method), 221 method), 730  |
| set_object() (AFL.automation.instrument.SpecScreen_Dsietr.SpejeScreen_APriliventomation.sample_env.TemperatureDeck.Driver  |
| method), 223 method), 733  |

912 Index

 $\verb|set_object(|)| (AFL. automation. loading. Load Stopper Driv \textit{set}(|)| (AFL. automation. sample\_env. Temperature Deck. Temperature Dec$ 

set\_object() (AFL.automation.loading.LoadStopperDrivsetDrivatput() (AFL.automation.shared.DataLabelerWidget.OrdinalEncod

 $\verb|set_object(|)| (AFL. automation. loading. Load Stopper Driv \verb|set_oad Stopper Driv \verb|set_oad Stopper Driv \verb|set_oad Stopper Driv set_oad Stopper Driv se$ 

*method*), 736

*method*), 746

| method), 767  | method), 278   |
|---|--|
| ${\sf set\_P()}\ (AFL. automation. loading. Digital Out Pressure Constitution (AFL) and the second contract of the s$ | nt <b>sehlersDigritæl OutAFelsxunteGrattvollko</b> ading.PneumaticPressureSampleCell.  |
| method), 37, 235, 236   | method), 283   |
| $\mathtt{set\_P()}$ (AFL.automation.loading.UltimusVPressureCont  | r <b>sletr_BetinplieVP (ess lineaCtontaction.</b> loading.PneumaticPressureSampleCell. |
| method), 60, 376, 377   | method), 288   |
| set_params()(AFL.automation.shared.DataLabelerWidg  | gesterdiampline (aleAFL.automation.loading.PneumaticSampleCell.Driver                  |
| method), 747  | method), 293   |
|   | er <b>s@tdisadtplcode</b> (AFL.automation.loading.PneumaticSampleCell.Pneumat          |
| method), 768  | method), 297   |
| set_properties_from_dict()  | set_sample()(AFL.automation.loading.PushPullSelectorSampleCell.Du                      |
| (AFL.automation.prepare.factory.Solution  | method), 308   |
| method), 715  | set_sample()(AFL.automation.loading.PushPullSelectorSampleCell.Pu                      |
| set_properties_from_dict()  | method), 313   |
|   | <pre>set_sample() (AFL.automation.loading.RSoXSSolutionSampleCell.Driv</pre>           |
| 391   | method), 319   |
|   | sss <b>ehiesเลต์ทุ่นะ()</b> (AFL.automation.loading.RSoXSSolutionSampleCell.RSo2       |
| method), 27, 822, 823   | method), 324   |
|   | set_sample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell                      |
| AFL.automation.APIServer.APIServer), 96   | method), 365   |
|   | set_sample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell                      |
| method), 22, 168, 171   | method), 371   |
|   | r. Betvesample() (AFL. automation. prepare. Dummy_OT2_Driver. Driver                   |
| method), 175  | method), 474   |
|   | r. <b>Dertrus plapilie()</b> (AFL.automation.prepare.Dummy_0T2_Driver.Dummy_0          |
| method), 179  | method), 478   |
|   | riset.Daimple() (AFL.automation.sample.CastingServer.CastingServer                     |
| method), 182  | method), 721   |
|   | riset_Bampilye(i)veAFL.automation.sample.CastingServer.Driver                          |
| method), 185  | method), 726   |
|   | Orsietr_sample() (AFL.automation.sample_env.TemperatureDeck.Driver                     |
| method), 195  | method), 733   |
|   | Dispersive Sansple() (AFL. automation. sample_env. Temperature Deck. Temperat          |
| method), 198  | method), 736   |
| set_sample()(AFL.automation.instrument.I22SAXS.Driv   |  |
| method), 201  | (AFL.automation.loading.PneumaticPressureSampleCell.Pneum                              |
| set_sample()(AFL.automation.instrument.I22SAXS.I22S   |  |
| method), 204  | set_sensor_config()  |
|   | Vis.Driver (AFL.automation.loading.PneumaticSampleCell.PneumaticSam                    |
| method), 207  | method), 45, 297, 301  |
| $set\_sample()$ (AFL.automation.instrument.SeabreezeUV  |  |
| method), 218  | (AFL.automation.APIServer.APIServer.ServiceInfo  |
| set_sample()(AFL.automation.instrument.SpecScreen_L   | · · · · · · · · · · · · · · · · · · ·  |
| method), 221  | set_server_if_missing()  |
|   | Driver.Spec <b>S&amp;Feknutoniver</b> ion.shared.ServerDiscovery.AsyncServiceInfo      |
| method), 223  | method), 784   |
| set_sample()(AFL.automation.loading.LoadStopperDri  |  |
| method), 248  | (AFL.automation.shared.ServerDiscovery.ServiceInfo                                     |
| set_sample() (AFL.automation.loading.LoadStopperDri   | · · · · · · · · · · · · · · · · · · ·  |
| method), 252  | set_shake() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O'                          |
| set_sample() (AFL.automation.loading.OneSelectorBlov  |  |
| method), 269  | set_shaker_temp()(AFL.automation.prepare.Dummy_OT2_Driver.Du.                          |
| set_sample() (AFL.automation.loading.OneSelectorBlov  |  |
| method) 274   | set state()(AFL automation prepare DeckRuilderWidget Rutton                            |

set\_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell

- set\_state() (AFL.automation.prepare.DeckBuilderWidgetsetbesklatte() (AFL.automation.prepare.SweepBuilderWidget.Label method), 414 method), 681
- set\_state() (AFL.automation.prepare.DeckBuilderWidgetberoptlawe() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 427 method), 691
- set\_state() (AFL.automation.prepare.DeckBuilderWidgets\( \mathbb{HB}\_{C}\)\( \text{state()} \) (AFL.automation.prepare.SweepBuilderWidget.Text method), 435 \qquad method), 701
- set\_state() (AFL.automation.prepare.DeckBuilderWidgetsetate() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 443 method), 709
- set\_state() (AFL.automation.prepare.DeckBuilderWidgetsEntycourget() (AFL.automation.prepare.MassBalance method), 453 method), 385
- set\_state() (AFL.automation.prepare.DeckBuilderWidges Etxttemp() (AFL.automation.prepare.Dummy\_OT2\_Driver.Dummy\_OT2\_method), 461 method), 66, 477, 479

set\_state() (AFL.automation.prepare.DeckBuilderWidges &Botemp() (AFL.automation.sample\_env.TemperatureDeck.TemperatureD

- method), 469 method), 75, 735, 736
  set\_state() (AFL.automation.prepare.PrepareWidget.Button
- set\_state() (AFL.automation.prepare.PrepareWidget.Buttert\_trait() (AFL.automation.prepare.DeckBuilderWidget.Button method), 498 method), 406
- ${\tt set\_state()}~(AFL. automation. prepare. Prepare Widget. Checkbox method), 506 \\ method), 414$
- set\_state() (AFL.automation.prepare.PrepareWidget.Dro**get**b\_wtrait() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 516 method), 427
- set\_state() (AFL.automation.prepare.PrepareWidget.HBset\_trait() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 524 method), 435
- set\_state() (AFL.automation.prepare.PrepareWidget.Layset\_trait() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 542 method), 453
- set\_state() (AFL.automation.prepare.PrepareWidget.VB&et\_trait() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 563 method), 469
- set\_state() (AFL.automation.prepare.SampleSeriesWidgeteCheckbox () (AFL.automation.prepare.PrepareWidget.Checkbox method), 580 method), 506
- set\_state() (AFL.automation.prepare.SampleSeriesWidgeteFlortrEixt() (AFL.automation.prepare.PrepareWidget.Dropdown method), 591 method), 516
- set\_state() (AFL.automation.prepare.SampleSeriesWidgetHBbrait() (AFL.automation.prepare.PrepareWidget.HBox method), 599 method), 524
- set\_state() (AFL.automation.prepare.SampleSeriesWidgesehufTexait() (AFL.automation.prepare.PrepareWidget.Label method), 607 method), 532
- set\_state() (AFL.automation.prepare.SampleSeriesWidgesetaltedait() (AFL.automation.prepare.PrepareWidget.Layout method), 615 method), 542
- set\_state() (AFL.automation.prepare.SampleSeriesWidgesettaytemait() (AFL.automation.prepare.PrepareWidget.Text method), 625 method), 555
- set\_state() (AFL.automation.prepare.SampleSeriesWidgetElextrait() (AFL.automation.prepare.PrepareWidget.VBox method), 636
  method), 563
- ${\tt set\_state()}~(AFL. automation. prepare. Sample Series Widges {\tt exait()}~(AFL. automation. prepare. Sample Series Widget. Button method), 644 \\ method), 572$
- set\_state() (AFL.automation.prepare.SweepBuilderWidgseButtvait() (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 657 method), 580
- set\_state() (AFL.automation.prepare.SweepBuilderWidg**seCherkio**tx() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 665 method), 591
- set\_state() (AFL.automation.prepare.SweepBuilderWidgsetHBcmait() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 673 method), 599

```
set_trait() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 255
                                 method), 607
                                                                                                                                                                                                                setDaemon() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv)
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Label method), 258
                                                                                                                                                                                                                setDaemon() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
                                  method), 615
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 261
                                                                                                                                                                                                                \mathtt{setDaemon()}\ (AFL. automation. loading. Sensor. Dummy Sensor 1
                                 method), 625
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Text
                                                                                                                                                                                                                                                 method), 335
                                 method), 636
                                                                                                                                                                                                                 setDaemon() (AFL.automation.loading.Sensor.DummySensor2
set_trait() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 339
                                 method), 644
                                                                                                                                                                                                                setDaemon() (AFL. automation. loading. Sensor Callback Thread. Sensor Callb
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Button method), 344
                                                                                                                                                                                                                 setDaemon() (AFL.automation.loading.SensorCallbackThread.SimpleThre
                                 method), 657
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Checkboxethod), 346
                                 method), 665
                                                                                                                                                                                                                setDaemon() (AFL.automation.loading.SensorCallbackThread.StopLoadC
set_trait() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 350
                                                                                                                                                                                                                 setDaemon() (AFL.automation.loading.SensorCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThread.StopLoadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThreadCallbackThrea
                                  method), 673
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Label method), 353
                                                                                                                                                                                                                setDaemon() (AFL.automation.loading.SensorPollingThread.SensorPollin
                                  method), 681
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 358
                                 method), 691
                                                                                                                                                                                                                 setDaemon() (AFL.automation.shared.ServerDiscovery.RunThread
set\_trait() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                                                                                                                                                 method), 791
                                                                                                                                                                                                                \mathtt{setDaemon()}\ (AFL. automation. shared. Server Discovery. Service Browser
                                 method), 701
set_trait() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 796
                                  method), 709
                                                                                                                                                                                                                setdefault() (AFL.automation.APIServer.Driver.PersistentConfig
set_units_cb() (AFL.automation.prepare.PrepareWidget.StockBuilthentWidget.71
                                 method), 548
                                                                                                                                                                                                                setdefault() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
set_units_cb() (AFL.automation.prepare.StockBuilderWidget.StockBathldd)Widget
                                 method), 72, 648, 650
                                                                                                                                                                                                                setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. One Selector Blowout Sample Cell. and the setdefault() (AFL. automation. loading. Automation. Auto
set_volume() (AFL.automation.prepare.Component.Component
                                                                                                                                                                                                                                                  method), 280
                                 method), 64, 396, 399
                                                                                                                                                                                                                setdefault() (AFL. automation. loading. Pneumatic Pressure Sample Cell. default()
set_volume() (AFL.automation.prepare.factory.Solution
                                                                                                                                                                                                                                                   method), 290
                                 method), 715
                                                                                                                                                                                                                setdefault() (AFL. automation. loading. Pneumatic Sample Cell. default disconnected by the set of the set o
set_volume()
                                                                                      (AFL.automation.prepare.Solute
                                                                                                                                                                                                                                                   method), 300
                                  method), 390
                                                                                                                                                                                                                setdefault() (AFL. automation. loading. PushPullSelectorSampleCell. default()
                                                                               (AFL.automation.prepare.Solution
                                                                                                                                                                                                                                                   method), 316
set_volume()
                                 method), 392
                                                                                                                                                                                                                setdefault() (AFL.automation.loading.RSoXSSolutionSampleCell.default
set_volume()
                                                                                  (AFL.automation.prepare.Solvent
                                                                                                                                                                                                                                                  method), 327
                                 method), 394
                                                                                                                                                                                                                setdefault() (AFL. automation. loading. Two Selector Blowout Sample Cell. at the set of the set 
setAllChannelsOff()
                                                                                                                                                                                                                                                   method), 373
                                 (AFL.automation.loading.SainSmartRelay.SainSmartRelay.Local (AFL.automation.shared.DatasetWidget.defaultdict
                                 method), 51, 331
                                                                                                                                                                                                                                                  method), 756
setChannels() (AFL.automation.loading.MultiChannelRetactMultiChannelRetactMultiChannels()
                                 method), 40, 262, 263
                                                                                                                                                                                                                                                  method), 772
setChannels() (AFL.automation.loading.SainSmartRelay. Methits Have the (Bethits L.automation.shared.PersistentConfig.PersistentConfig.
                                 method), 330
                                                                                                                                                                                                                                                   method), 775
setChannels() (AFL.automation.loading.SainSmartRelay.SetDShlaytRelahFL.automation.loading.ChemyxSyringePump.ChemyxConn
```

setDaemon() (AFL.automation.APIServer.APIServer.QueusDtDirameter() (AFL.automation.loading.ChemyxSyringePump.ChemyxC

setDaemon() (AFL.automation.APIServer.QueueDaemon.QueuExDoosnume() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeU

setDaemon() (AFL.automation.EpicsADLiveProcess.ReducseDaenporsuReeDeaDyn(AFL.automation.instrument.SeabreezeUVVis.Seab

method), 36, 230, 234

method), 36, 230, 234

method), 30, 217, 218

method), 30, 216, 218

Index 915

method), 51, 331, 332

method), 144

*method*), 191

*method*), 826

```
method), 30, 216, 218
                                                                                                                                                                                                                    method), 242
setFilepath() (AFL.automation.instrument.SeabreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeUV\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\setSeatbreezeU\set
                             method), 30, 216, 218
                                                                                                                                                                                                                    method), 41, 265, 267
setFormatter() (AFL.automation.APIServer.APIServer.FiseMRatl&G) (AFL.automation.loading.NE1kSyringePump.SyringePump
                             method), 105
                                                                                                                                                                                                                     method), 266
setFormatter() (AFL.automation.APIServer.APIServer.SMFTRIAmalle(AFL.automation.loading.PressureControllerAsPump.Pressure
                                                                                                                                                                                                                    method), 46, 303, 305
                             method), 147
setLevel() (AFL.automation.APIServer.APIServer.FileHandtate() (AFL.automation.loading.PressureControllerAsPump.SyringeF
                              method), 105
                                                                                                                                                                                                                     method), 305
setLevel() (AFL.automation.APIServer.APIServer.SMTPHentilate() (AFL.automation.loading.SyringePump.SyringePump
                              method), 147
                                                                                                                                                                                                                     method), 58, 361
setLevel() (AFL.automation.loading.ChemyxSyringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpsc
                             method), 35, 231, 233
                                                                                                                                                                                                                     method), 273
setName() (AFL.automation.APIServer.APIServer.QueueDsetRimseLevel() (AFL.automation.loading.OneSelectorBlowoutSampleC
                             method), 144
                                                                                                                                                                                                                     method), 277
setName() (AFL.automation.APIServer.QueueDaemon.QuesexRamselvevel() (AFL.automation.loading.PneumaticPressureSampleCe
                             method), 191
                                                                                                                                                                                                                     method), 44, 286, 291
setName() (AFL.automation.EpicsADLiveProcess.ReduceDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedveeDarRimsRedveeDa
                                                                                                                                                                                                                     method), 45, 296, 301
                             method), 826
setName() (AFL.automation.loading.LoadStopperDriver.ScstorPolialgeVall() (AFL.automation.loading.RSoXSSolutionSampleCell.R.
                             method), 255
                                                                                                                                                                                                                     method), 50, 323, 328
setName() (AFL.automation.loading.LoadStopperDriver.StoptRind&Budvel() (AFL.automation.loading.TwoSelectorBlowoutSampleC
                                                                                                                                                                                                                     method), 60, 370, 374
                              method), 258
setName() (AFL.automation.loading.LoadStopperDriver.StoptSauleStant()
                             method), 261
                                                                                                                                                                                                                     (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis
setName() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                     method), 30, 216, 218
                                                                                                                                                                                       \mathtt{setStream}() (AFL. automation. APIServer. APIServer. File Handler
                              method), 336
setName() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                     method), 105
                                                                                                                                                                                       setTime() (AFL.automation.loading.ChemyxSyringePump.ChemyxConnec
                             method), 339
setName() (AFL.automation.loading.SensorCallbackThread.SensorGaldthackThreadB0, 234
                              method), 344
                                                                                                                                                                                       setUnits() (AFL.automation.loading.ChemyxSyringePump.ChemyxConn
setName() (AFL.automation.loading.SensorCallbackThread.SimpleThreatModIdCB, 230, 234
                             method), 347
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Button
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadh@Blod), 406
                              method), 350
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Checkb
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadicaRv21), 414
                             method), 353
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Dropdo
setName() (AFL.automation.loading.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollingThread.SensorPollin
                              method), 358
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.HBox
setName() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                    method), 435
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Label
                             method), 791
setName() (AFL.automation.shared.ServerDiscovery.ServiceBrowsemethod), 443
                             method), 796
                                                                                                                                                                                      setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Layout
setParameter() (AFL.automation.loading.PushPullSelectorSample@edilh@dylh#ภีมิโSelectorSampleCell
                             method), 48, 311, 317
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Text
setRate() (AFL.automation.loading.ChemyxSyringePump.ChemyxCovenleadion461
                              method), 36, 230, 234
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.VBox
setRate() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringePump.ChemyxSyringeP
                              method), 35, 231, 233
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.PrepareWidget.Button
setRate() (AFL.automation.loading.ChemyxSyringePump.SyringePumphod), 498
                             method), 232
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.PrepareWidget.Checkbox
setRate() (AFL.automation.loading.DummyPump.DummyPump
                                                                                                                                                                                                                    method), 506
                              method), 38, 241, 243
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.PrepareWidget.Dropdown
```

method), 516

setRate() (AFL.automation.loading.DummyPump.SyringePump

```
setup_instance() (AFL.automation.prepare.PrepareWidsatMBsteLevel() (AFL.automation.loading.RSoXSSolutionSampleCell.R.
               method), 524
                                                                                                          method), 50, 324, 328
setup_instance() (AFL.automation.prepare.PrepareWidgettNidgettelevel() (AFL.automation.loading.TwoSelectorBlowoutSampleC
                                                                                                          method), 60, 371, 374
               method), 532
setup_instance() (AFL.automation.prepare.PrepareWidghteldy_ocontext_processor()
              method), 542
                                                                                                          (AFL.automation.APIServer.APIServer.Flask
setup_instance() (AFL.automation.prepare.PrepareWidget.Text method), 122
               method), 555
                                                                                           shell_context_processors
setup_instance() (AFL.automation.prepare.PrepareWidget.VBox (AFL.automation.APIServer.APIServer.Flask
              method), 563
                                                                                                          attribute), 114
setup_instance() (AFL.automation.prepare.SampleSeriesHouled_Raymore_error()
                                                                                                          (AFL.automation.APIServer.APIServer.Flask
               method), 572
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.Checkbd), 124
              method), 580
                                                                                           shuffle()
                                                                                                                      (AFL.automation.prepare.SampleSeries
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.FiboethRed), 388
               method), 591
                                                                                           SimpleThresholdCB
                                                                                                                                                  (class
                                                                                                                                                                                in
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.HANDL.automation.loading.SensorCallbackThread),
              method), 599
                                                                                                          55, 344, 355
setup_instance() (AFL.automation.prepare.SampleSeries Widge(AHILexattomation.prepare.factory.Solution prop-
              method), 607
                                                                                                          erty), 715
setup_instance() (AFL.automation.prepare.SampleSeries Widget #Fahadtomation.prepare.Solution property), 391
              method), 615
                                                                                           sld
                                                                                                      (AFL.automation.prepare.Component.Component
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.Lpxoperty), 64, 396, 399
                                                                                           sld (AFL.automation.prepare.Solute property), 389
               method), 625
setup_instance() (AFL.automation.prepare.SampleSeries Widge Hawtomation.prepare.Solvent property), 393
              method), 636
                                                                                           SMTPHandler
                                                                                                                                            (class
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.V&BL.automation.APIServer.APIServer), 145
                                                                                           Solute (AFL.automation.prepare.Component.PrepType
              method), 644
setup_instance() (AFL.automation.prepare.SweepBuilderWidget.Buttrohute), 397
              method), 657
                                                                                           Solute (AFL.automation.prepare.PrepType.PrepType at-
setup_instance() (AFL.automation.prepare.SweepBuilderWidget.Ghibaktleox68, 490, 491
               method), 665
                                                                                           Solute (class in AFL.automation.prepare), 388
setup_instance() (AFL.automation.prepare.SweepBuildestWidgesHBox (AFL.automation.prepare.factory.Solution
              method), 673
                                                                                                          property), 715
setup_instance() (AFL.automation.prepare.SweepBuildestVindpesLautomation.prepare.Solution property),
              method), 681
setup_instance() (AFL.automation.prepare.SweepBuild&dViidgiohdAfilt.automation.prepare.Component.PrepType
               method), 691
                                                                                                          attribute), 397
setup_instance() (AFL.automation.prepare.SweepBuild&WWdgioTe(AFL.automation.prepare.PrepType.PrepType
              method), 701
                                                                                                          attribute), 68, 490, 491
setup_instance() (AFL.automation.prepare.SweepBuild&dVindgioVIBobass in AFL.automation.prepare), 390
              method), 709
                                                                                           Solution (class in AFL.automation.prepare.factory),
setupDefaults() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
                                                                                           Solvent (AFL.automation.prepare.Component.PrepType
              method), 189
setVolume() (AFL.automation.loading.ChemyxSyringePump.ChemyaCribmae);@97
               method), 36, 230, 234
                                                                                           Solvent (AFL.automation.prepare.PrepType.PrepType
setWasteLevel() (AFL.automation.loading.OneSelectorBlowoutSamphiEvelk)() (AFL.automation.loading.OneSelector
                                                                                           Solvent (class in AFL.automation.prepare), 392
              method), 274
setWasteLevel() (AFL.automation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandationation.loading.OneSelectorBioNometSundianStandation.
               method), 278
                                                                                                          property), 715
setWasteLevel() (AFL.automation.loading.PneumaticPresolveSaturgeleGslitPyneumAdIc.RutosmurtiSompregatel.Solution
                                                                                                          property), 392
              method), 44, 287, 291
setWasteLevel() (AFL.automation.loading.PneumaticSampleCedtPnassutAiFSampleCedtlon.prepare.factory.Solution
               method), 45, 296, 301
                                                                                                          property), 715
```

- solvent\_mass (AFL.automation.prepare.Solution property), 392
- solvent\_sld (AFL.automation.prepare.factory.Solution property), 715
- solvent\_sld (AFL.automation.prepare.Solution property), 392
- $\verb"solvent_volume" (AFL. automation. prepare. factory. Solution$ property), 715 start() (AFL.automation.loading.SensorCallbackThread.SensorCallback
- solvent\_volume (AFL.automation.prepare.Solution property), 392
- solvents (AFL.automation.prepare.factory.Solution property), 715
- solvents (AFL.automation.prepare.Solution property), 391
- SpecScreen\_Driver (class in
- AFL.automation.instrument.SpecScreen\_Driver), start() (AFL.automation.loading.SensorPollingThread.SensorPo 31, 222, 223
- split\_vars() (AFL.automation.shared.DatasetWidget.DatsstatWidgetAMLoahelomation.prepare.DeckBuilderWidget.DeckBuilderWidget method), 79, 752, 758 method), 65, 420, 471

method), 258

method), 261

method), 336

*method*), 339

method), 344

method), 347

method), 350

*method*), 353

method), 358

method), 65, 421, 471

method), 69, 545, 565

start() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2

 ${\tt start()}\ (AFL. automation. loading. Sensor Callback Thread. Simple Threshold the Computation of the Co$ 

start() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1

start() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2

start() (AFL.automation.loading.Sensor.DummySensor1

start() (AFL.automation.loading.Sensor.DummySensor2

- sqrt() (in module AFL.automation.APIServer.Driver), start() (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget
- $\verb|sqrt()| (in module AFL. automation. APIS erver. Dummy Driver)| (AFL. automation. prepare . Prepare Widget. Deck Builder Widget . Deck Builder . Deck Buil$ method), 509
- sqrt() (in module AFL.automation.APIServer.DummyOT2Btimet)() (AFL.automation.prepare.PrepareWidget.PrepareWidget
- method), 69, 544, 565 sqrt() (in module AFL.automation.instrument.SpecScreen\_Daret() (AFL.automation.prepare.PrepareWidget.PrepareWidget\_View
- sqrt() (in module AFL.automation.prepare.DeckBuilderWidget); () (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget method), 547
- $\mathtt{sqrt}()$  (in module AFL.automation.prepare.PrepareWidgetStart() (AFL.automation.prepare.PrepareWidget.StockBuilderWidgetStart()) method), 548
- sqrt() (in module AFL.automation.prepare.SampleSeriesWidget),() (AFL.automation.prepare.PrepareWidget.SweepBuilderWidget method), 549
- sqrt() (in module AFL.automation.prepare.StockBuilderWistgen); () (AFL.automation.prepare.SampleSeriesWidget method), 71, 629, 646
- sqrt() (in module AFL.automation.prepare.SweepBuilderWsitleart,() (AFL.automation.prepare.SampleSeriesWidget method), 71, 630, 647
- sqrt() (in module AFL.automation.sample.CastingServer), start() (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget. method), 72, 648, 650
- sqrt() (in module AFL.automation.shared.DataLabelerWidgetStrt() (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget. method), 72, 649, 650
- sqrt() (in module AFL.automation.shared.DiffractionLabekstart() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilderwoopBuilderwidget.SweepBuilderwidget.SweepBuilderwidget.SweepBuil method), 73, 694, 711  $\verb|start()| (AFL. automation. APIS erver. APIS erver. Queue Dae \textit{\texttt{Broart}()}) (AFL. automation. prepare. Sweep Builder Widget. Sw$
- method), 144 method), 73, 695, 711 start() (AFL.automation.APIServer.APIServer.Zeroconf start() (AFL.automation.shared.ServerDiscovery.RunThread
- method), 154 *method*), 791
- start() (AFL.automation.APIServer.QueueDaemon.QueueDaemoh() (AFL.automation.shared.ServerDiscovery.ServiceBrowser method), 796 *method*), 192
- start() (AFL.automation.EpicsADLiveProcess.ReduceDaestant.ReduceDa method), 826 method), 805
- start() (AFL.automation.loading.LoadStopperDriver.SensotRanland(AFLautomation.APIServer.APIServer.Zeroconf method), 255 property), 154
- start() (AFL.automation.loading.LoadStopperDriver.Stop&badC&b(AFL.automation.shared.ServerDiscovery.Zeroconf

```
property), 805
                                                                                                                                                                                                                                                                 method), 45, 296, 300
startPump() (AFL.automation.loading.ChemyxSyringePungpathesn()x(Childrentionnation.loading.PushPullSelectorSampleCell.Driver
                                    method), 36, 230, 233
                                                                                                                                                                                                                                                                 method), 308
stat() (AFL.automation.instrument.SeabreezeUVVis.Path status() (AFL.automation.loading.PushPullSelectorSampleCell.PushPull
                                    method), 211
                                                                                                                                                                                                                                                                  method), 48, 311, 317
static_folder(AFL.automation.APIServer.APIServer.Flastatus() (AFL.automation.loading.RSoXSSolutionSampleCell.Driver
                                                                                                                                                                                                                                                                  method), 319
                                    property), 129
static_url_path(AFL.automation.APIServer.APIServer.Etwatus()(AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolu
                                    property), 129
                                                                                                                                                                                                                                                                  method), 50, 322, 328
status()
                                                       (AFL.automation.APIServer.Driver.Driver status() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.Drive
                                    method), 22, 168, 172
                                                                                                                                                                                                                                                                  method), 365
status() (AFL.automation.APIServer.DummyDriver.Drivestatus() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.TwoSelectorBlowotSampleCell.Two
                                    method), 175
                                                                                                                                                                                                                                                                  method), 59, 369, 374
status() (AFL.automation.APIServer.DummyDriver.DummyDriver.Dummy_OT2_Driver.Driver
                                    method), 23, 178, 179
                                                                                                                                                                                                                                                                  method), 474
status() (AFL.automation.APIServer.DummyOT2Driver.Driver.Univer.Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dum
                                    method), 182
                                                                                                                                                                                                                                                                  method), 66, 477, 479
status() (AFL.automation.APIServer.DummyOT2Driver.DatawayDf)vAFL.automation.sample.CastingServer.CastingServer
                                    method), 24, 184, 186
                                                                                                                                                                                                                                                                  method), 89, 720, 730
status() (AFL.automation.EpicsADLiveProcess.AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxLivs(AreaDetestoxL
                                    method), 26, 821
                                                                                                                                                                                                                                                                  method), 726
status() (AFL.automation.instrument.DummySAS.Driver status() (AFL.automation.sample_env.TemperatureDeck.Driver
                                    method), 195
                                                                                                                                                                                                                                                                  method), 733
status() (AFL.automation.instrument.DummySAS.DummyS4stus() (AFL.automation.sample_env.TemperatureDeck.TemperatureDeck
                                    method), 28, 197, 198
                                                                                                                                                                                                                                                                 method), 75, 735, 736
status() (AFL.automation.instrument.I22SAXS.Driver stem (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                                                                                 property), 214
                                    method), 201
status() (AFL.automation.instrument.I22SAXS.I22SAXS step (AFL.automation.prepare.SampleSeriesWidget.FloatText
                                                                                                                                                                                                                                                                  attribute), 587
                                    method), 204
status() (AFL.automation.instrument.SeabreezeUVVis.Dristeep (AFL.automation.prepare.SampleSeriesWidget.IntText
                                    method), 207
                                                                                                                                                                                                                                                                  attribute), 603
status()(AFL.automation.instrument.SeabreezeUVVis.Se&browekBUiVider_reset_cb()
                                    method), 218
                                                                                                                                                                                                                                                                  (AFL.automation.prepare.PrepareWidget.PrepareWidget
                                                                                                                                                                                                                                                                 method), 69, 544, 565
status() (AFL.automation.instrument.SpecScreen_Driver.Driver
                                    method), 221
                                                                                                                                                                                                                            StockBuilderWidget
                                                                                                                                                                                                                                                                                                                                                                    (class
                                                                                                                                                                                                                                                                                                                                                                                                                                         in
status() (AFL.automation.instrument.SpecScreen_Driver.SpecScreeAFDnivetomation.prepare.PrepareWidget),
                                    method), 31, 222, 224
status()(AFL.automation.loading.LoadStopperDriver.Dr&tockBuilderWidget
                                                                                                                                                                                                                                                                                                                                                                    (class
                                                                                                                                                                                                                                                                                                                                                                                                                                         in
                                    method), 248
                                                                                                                                                                                                                                                                 AFL.automation.prepare.StockBuilderWidget),
status() (AFL.automation.loading.LoadStopperDriver.LoadStopperDrive47, 649
                                                                                                                                                                                                                            StockBuilderWidget_Model
                                    method), 39, 251, 262
                                                                                                                                                                                                                                                                                                                                                                                  (class
                                                                                                                                                                                                                                                                                                                                                                                                                                         in
status() (AFL.automation.loading.OneSelectorBlowoutSampleCell.PFikarutomation.prepare.StockBuilderWidget),
                                                                                                                                                                                                                                                                  72, 649, 650
                                    method), 269
status() (AFL.automation.loading.OneSelectorBlowoutSastpletBilliOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOn
                                                                                                                                                                                                                                                                                                                                                                                                                                         in
                                    method), 274
                                                                                                                                                                                                                                                                 AFL.automation.prepare.StockBuilderWidget),
Status() (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                                            \verb|stop()| (AFL. automation. loading. Chemyx Syringe Pump. Chemyx Syrin
                                    method), 277
status() (AFL.automation.loading.PneumaticPressureSampleCell.Dniethrod), 35, 231, 233
                                    method), 283
                                                                                                                                                                                                                            stop() (AFL.automation.loading.ChemyxSyringePump.SyringePump
status() (AFL.automation.loading.PneumaticPressureSampleCell.Pmeuhovat); PressureSampleCell
                                    method), 43, 286, 290
                                                                                                                                                                                                                             stop() (AFL. automation. loading. Digital Out Pressure Controller. Digital Out Pressure Controlle
status() (AFL.automation.loading.PneumaticSampleCell.Driver
                                                                                                                                                                                                                                                           method), 235
```

status() (AFL.automation.loading.PneumaticSampleCell.PneumatiosethpleCell6

stop() (AFL.automation.loading.DigitalOutPressureController.PressureC

*method*), 293

attribute), 457

```
stop() (AFL.automation.loading.DummyPump.DummyPumpyPumpyPum (AFL.automation.prepare.PrepareWidget.Button
                                       method), 38, 241, 242
                                                                                                                                                                                                                                                                                   attribute), 494
stop() (AFL.automation.loading.DummyPump.SyringePunstyle (AFL.automation.prepare.PrepareWidget.Checkbox
                                       method), 242
                                                                                                                                                                                                                                                                                   attribute), 502
stop() (AFL:automation.loading.NE1kSyringePump.NE1kSyringe(Puffipautomation.prepare.PrepareWidget.Dropdown
                                      method), 41, 264, 267
                                                                                                                                                                                                                                                                                  attribute), 516
stop() (AFL.automation.loading.NE1kSyringePump.SyringerPythep
                                                                                                                                                                                                                                                                                 (AFL.automation.prepare.PrepareWidget.Label
                                                                                                                                                                                                                                                                                   attribute), 528
                                       method), 266
stop() (AFL:automation.loading.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.PressureController.P
                                       method), 46, 302
                                                                                                                                                                                                                                                                                   tribute), 551
stop() (AFL.automation.loading.PressureControllerAsPungt.PressAreContoollatieAssPonepoure.SampleSeriesWidget.Button
                                       method), 46, 303, 305
                                                                                                                                                                                                                                                                                   attribute), 568
stop() (AFL.automation.loading.PressureControllerAsPunst$\infty\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\delta\de
                                       method), 304
                                                                                                                                                                                                                                                                                   attribute), 576
\verb|stop()| (AFL. automation. loading. Syringe Pump. Syringe Pump \verb|style| (AFL. automation. prepare. Sample Series Widget. Float Text) | The state of the state 
                                       method), 57, 361
                                                                                                                                                                                                                                                                                   attribute), 591
stop() (AFL.automation.loading.UltimusVPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureCont
                                       method), 375
                                                                                                                                                                                                                                                                                  attribute), 607
stop() (AFL.automation.loading.UltimusVPressureControlsaryWhithAFLYMhrassuntiGupmaplure.SampleSeriesWidget.Label
                                       method), 377
                                                                                                                                                                                                                                                                                   attribute), 611
stop_shake() (AFL.automation.prepare.Dummy_OT2_Drivey.DataFils_.Off@mDatione.prepare.SampleSeriesWidget.Text
                                      method), 66, 477, 479
                                                                                                                                                                                                                                                                                  attribute), 632
stopLoad() (AFL.automation.loading.PneumaticPressureSstrypleC&ITLP.nationnaticPressureSatisplecGBlitlderWidget.Button
                                       method), 43, 286, 291
                                                                                                                                                                                                                                                                                   attribute), 653
stopLoad() (AFL:automation.loading.PneumaticSampleCest.Pheu(AhFliaSutoupketGent)prepare.SweepBuilderWidget.Checkbox
                                       method), 45, 296, 300
                                                                                                                                                                                                                                                                                  attribute), 661
stopLoad() (AFL.automation.loading.SensorCallbackThrewithDau/dkFCanthonmatiatincpurepare.SweepBuilderWidget.Label
                                      method), 56, 341, 355
                                                                                                                                                                                                                                                                                   attribute), 677
StopLoadCBv1
                                                                                                                                (class
                                                                                                                                                                                                                                         style (AFL.automation.prepare.SweepBuilderWidget.Text
                                      AFL.automation.loading.LoadStopperDriver),
                                                                                                                                                                                                                                                                                   attribute), 697
                                       255
                                                                                                                                                                                                                                           submit_cb() (AFL.automation.prepare.PrepareWidget.SampleSeriesWidg
StopLoadCBv1
                                                                                                                                (class
                                                                                                                                                                                                                          in
                                                                                                                                                                                                                                                                                  method), 546
                                      AFL.automation.loading.SensorCallbackThread), submit_cb() (AFL.automation.prepare.SampleSeriesWidget.SampleSeries
                                       54, 347, 354
                                                                                                                                                                                                                                                                                  method), 70, 628, 646
StopLoadCBv2
                                                                                                                                (class
                                                                                                                                                                                                                                          submit_mixing_wells_cb()
                                      AFL.automation.loading.LoadStopperDriver),
                                                                                                                                                                                                                                                                                  (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget
                                       258
                                                                                                                                                                                                                                                                                  method), 547
StopLoadCBv2
                                                                                                                                (class
                                                                                                                                                                                                                          in submit_mixing_wells_cb()
                                      AFL.automation.loading.SensorCallbackThread),
                                                                                                                                                                                                                                                                                  (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
                                       55, 350, 354
                                                                                                                                                                                                                                                                                  method), 71, 629, 646
stopPump() (AFL.automation.loading.ChemyxSyringePumpx(Hring)(AGhmeattionation.instrument.SeabreezeUVVis.Path
                                       method), 36, 230, 233
                                                                                                                                                                                                                                                                                 property), 214
                                                                                                                                                                                                                                         {\tt suffixes}\, (AFL. automation. instrument. Seabreeze UVV is. Path
strtobool()
                                                                                                                         (in
                                                                                                                                                                                                    module
                                      AFL.automation.APIServer.APIServer), 96
                                                                                                                                                                                                                                                                                 property), 214
\verb|style| (AFL. automation.prepare.DeckBuilderWidget.ButtonSweepBuilder\_reset\_cb()|
                                                                                                                                                                                                                                                                                   (AFL.automation.prepare.PrepareWidget.PrepareWidget
                                       attribute), 402
style(AFL.automation.prepare.DeckBuilderWidget.Checkbox)
                                                                                                                                                                                                                                                                                  method), 69, 544, 565
                                                                                                                                                                                                                                           SweepBuilderWidget
                                       attribute), 410
                                                                                                                                                                                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    in
style(AFL. automation. prepare. Deck Builder Widget. Dropdown
                                                                                                                                                                                                                                                                                 AFL.automation.prepare.PrepareWidget),
                                       attribute), 427
\verb|style| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|SweepBuilder Widget|)| | SweepBuilder Widget | SweepBuilder Widget | SweepBuilder Widget| |
                                                                                                                                                                                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                     in
                                                                                                                                                                                                                                                                                 AFL.automation.prepare.SweepBuilderWidget),
                                      attribute), 439
{\tt style}\,(AFL. automation. prepare. Deck Builder Widget. Text
                                                                                                                                                                                                                                                                                   73, 693, 711
```

920 Index

SweepBuilderWidget\_Model

(class

in

```
AFL.automation.prepare.SweepBuilderWidget), tabbable (AFL.automation.prepare.PrepareWidget.Button
                   73, 694, 711
                                                                                                                                       attribute), 498
SweepBuilderWidget_View
                                                                                                                   tabbable (AFL.automation.prepare.PrepareWidget.Checkbox
                  AFL. automation. prepare. Sweep Builder Widget),
                                                                                                                                       attribute), 506
                   73, 694, 711
                                                                                                                   tabbable (AFL.automation.prepare.PrepareWidget.Dropdown
swish() (AFL.automation.loading.OneSelectorBlowoutSampleCell.OntaBidlutatorBlowoutSampleCell
                   method), 274
                                                                                                                   tabbable (AFL.automation.prepare.PrepareWidget.HBox
swish() (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwttSiHutt@rBldwoutSampleCell
                   method), 277
                                                                                                                   tabbable (AFL.automation.prepare.PrepareWidget.Label
swish() (AFL.automation.loading.PushPullSelectorSampleCell.PushAttuliSule);t&tSampleCell
                   method), 48, 313, 317
                                                                                                                   tabbable (AFL.automation.prepare.PrepareWidget.Text
swish() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXS&ahibioreSanfileCell
                                                                                                                   {\tt tabbable} \ (AFL. automation. prepare. Prepare Widget. VBox
                  method), 50, 323, 328
swish() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwttSithateorBlowoutSampleCell
                   method), 60, 370, 374
                                                                                                                   {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. Button
symlink_to() (AFL.automation.instrument.SeabreezeUVVis.Path attribute), 572
                                                                                                                   {\tt tabbable} (AFL. automation. prepare. Sample Series Widget. Checkbox
                  method), 212
sync_to_prepare_cb()
                                                                                                                                      attribute), 580
                   (AFL.automation.prepare.PrepareWidget.SampleSexteds MilleAtAFL.automation.prepare.SampleSeriesWidget.FloatText
                  method), 547
                                                                                                                                       attribute), 591
sync_to_prepare_cb()
                                                                                                                   {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. HBox
                   (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget), 599
                  method), 70, 629, 646
                                                                                                                   {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. Int Text
SyringePump
                                                                                                                                       attribute), 607
                  AFL.automation.loading.ChemyxSyringePump), tabbable (AFL.automation.prepare.SampleSeriesWidget.Label
                   232
                                                                                                                                      attribute), 615
SyringePump
                                                             (class
                                                                                                                   \verb+tabbable+ (AFL. automation. prepare. Sample Series Widget. Text
                  AFL.automation.loading.DummyPump),
                                                                                                                                       attribute), 636
                                                                                                                   tabbable (AFL.automation.prepare.SampleSeriesWidget.VBox
SyringePump
                                                             (class
                                                                                                          in
                                                                                                                                      attribute), 644
                  AFL.automation.loading.NE1kSyringePump),
                                                                                                                   tabbable (AFL.automation.prepare.SweepBuilderWidget.Button
                   265
                                                                                                                                       attribute), 657
                                                                                                                  tabbable (AFL.automation.prepare.SweepBuilderWidget.Checkbox
SyringePump
                                                             (class
                                                                                                           in
                  AFL.automation.loading.PressureControllerAsPump),
                                                                                                                                      attribute), 665
                   304
                                                                                                                   tabbable (AFL.automation.prepare.SweepBuilderWidget.HBox
SyringePump
                                                             (class
                                                                                                                                      attribute), 673
                                                                                                          in
                  AFL.automation.loading.SyringePump),
                                                                                                        57,
                                                                                                                   tabbable (AFL.automation.prepare.SweepBuilderWidget.Label
                   361
                                                                                                                                       attribute), 681
                                                                                                                   tabbable (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                                      attribute), 701
{\tt tabbable} \ (AFL. automation. prepare. Deck Builder Widget. Buttot bable} \ (AFL. automation. prepare. Sweep Builder Widget. VBox) \ (AFL. automation. prepare. Sweep Builder Widget. Sweep Builder Wi
                                                                                                                                       attribute), 709
                   attribute), 406
tabbable (AFL. automation. prepare. Deck Builder Widget. Chroding the AFL. automation. prepare. Sample \ property), 387 and 387 are also below the automation of the AFL autom
                                                                                                                   target_check (AFL.automation.prepare.Sample prop-
                   attribute), 414
tabbable (AFL.automation.prepare.DeckBuilderWidget.Dropdown erty), 387
                                                                                                                   target_loc (AFL.automation.prepare.Sample prop-
                   attribute), 427
tabbable (AFL.automation.prepare.DeckBuilderWidget.HBox
                                                                                                                                      erty), 387
                                                                                                                   teardown_appcontext()
                   attribute), 435
{\tt tabbable} \ (AFL. automation. prepare. Deck Builder Widget. Label
                                                                                                                                      (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                      method), 121
                   attribute), 443
{\tt tabbable} \ (AFL. automation. prepare. Deck Builder Widget. \textit{Text} \textbf{mear} down\_app \textbf{context\_funcs}
                                                                                                                                      (AFL.automation.APIServer.APIServer.Flask
                   attribute), 461
                                                                                                                                      attribute), 114
tabbable (AFL.automation.prepare.DeckBuilderWidget.VBox
                                                                                                                   teardown_request() (AFL.automation.APIServer.APIServer.Flask
                   attribute), 469
```

| method), 130   | method), 119   |
|--|--|
| teardown_request_funcs   | test_cli_runner_class  |
| (AFL.automation.APIServer.APIServer.Flask  | (AFL.automation.APIServer.APIServer.Flask  |
| attribute), 131  | attribute), 113  |
| TemperatureDeck (class in  | ${\sf test\_client()}$ (AFL.automation.APIServer.APIServer.Flask   |
| AFL.automation.sample_env.TemperatureDeck),  | method), 118   |
| 75, 734, 736   | test_client_class(AFL.automation.APIServer.APIServer.Flask   |
| template_context_processors  | attribute), 113  |
| (AFL. automation. APIS erver. APIS erver. Flask  | $\verb test_command1()  (AFL. automation. APIS erver. Dummy Driver. Driver.$ |
| attribute), 131  | method), 23, 178, 179  |
| template_filter() (AFL.automation.APIServer.AP   | vtveEtc.skommand1() (AFL.automation.APIServer.DummyOT2Driver.Dummy<br>method), 24, 184, 186  |
| template_folder(AFL.automation.APIServer.APIServer   | :Fbst_command2() (AFL.automation.APIServer.DummyDriver.DummyDriv   |
| attribute), 130  | method), 23, 178, 179  |
| <pre>template_global() (AFL.automation.APIServer.APISer</pre>  | veteEtruscommand2()(AFL.automation.APIServer.DummyOT2Driver.Dummy  |
| method), 121   | method), 24, 184, 186  |
| <pre>template_test() (AFL.automation.APIServer.APIServer</pre>   | :Fksk_command_sets_data()  |
| method), 121   | (AFL.automation.APIServer.DummyDriver.DummyDriver  |
| templates_auto_reload  | method), 23, 178, 179  |
| (AFL.automation.APIServer.APIServer.Flask  | test_command_sets_data()   |
| property), 116   | (AFL.automation.APIServer.DummyOT2Driver.DummyDriver   |
| terminate()(AFL.automation.APIServer.APIServer.Quer  | ueDaemon method), 24, 184, 186   |
| method), 143   | $\verb test_image()  (AFL. automation. APIS erver. Dummy Driver. Dummy Driver) $   |
| terminate()(AFL.automation.APIServer.QueueDaemon.  | QueueDaemothod), 23, 178, 180  |
| method), 25, 190, 192  | $\verb test_image()  (AFL. automation. APIS erver. Dummy OT2 Driver. Dummy $ |
| ${\tt terminate()} \ (AFL. automation. Epics ADLive Process. Reduced the process and the process and the process and the process and the process are process. The process are process and the process are process and the process are process. The process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process. The process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process. The process are process are process are process are process are process are process. The process are process. The process are p$ |  |
| method), 27, 824, 826  | test_plot() (AFL.automation.APIServer.DummyDriver.DummyDriver  |
| ${\tt terminate()} \ (AFL. automation. loading. Load Stopper Drive and the properties of the properties$ |  |
| method), 254   | ${\sf test\_plot()}\ (AFL. automation. APIS erver. Dummy OT2 Driver. Dummy Driver. Driver.$  |
| ${\tt terminate()} \ (AFL. automation. loading. Load Stopper Drive and Stopper Drive $ |  |
| method), 258   | test_request_context()   |
| terminate() (AFL.automation.loading.LoadStopperDrive method), 261  | method), 133   |
| terminate()(AFL.automation.loading.Sensor.DummySen   | nstæsting (AFL.automation.APIServer.APIServer.Flask  |
| method), 53, 334, 340  | attribute), 111  |
| ${\tt terminate()} \ (AFL. automation. loading. Sensor. Dummy Sensor. Dum$ | nstælt (AFL.automation.APIServer.APIServer.ServiceInfo   |
| method), 53, 337, 340  | attribute), 149  |
| method), 54, 343, 353  | ar <b>eaxl:SeAFA:GalbhackThr:dual</b> red.ServerDiscovery.AsyncServiceInfo<br>attribute), 781  |
|  | ก <b>read:tS(AupleTittenhalidGB</b> hared.ServerDiscovery.ServiceInfo  |
| method), 347   | attribute), 799  |
| terminate() (AFL.automation.loading.SensorCallbackTh<br>method), 350   | u <b>read:S(ohlsoad&amp;BŁ</b> lautomation.prepare.DeckBuilderWidget),<br>455  |
| terminate() (AFL.automation.loading.SensorCallbackTh<br>method), 353   | ar <b>ead.tS(aphtxxailCBFL</b> .automation.prepare.PrepareWidget),<br>549  |
|  | e <b>Tde Str(sba-RobbiA&amp;Thwead</b> mation.prepare.SampleSeriesWidget),<br>630  |
|  | alidkt (class in AFL.automation.prepare.SweepBuilderWidget),   |
| attribute), 105  | 695  |
| ternary_click_callback()   | timed_dispense() (AFL.automation.loading.DigitalOutPressureControll  |
| (AFL.automation.shared.DiffractionLabeler.Diffr  |  |
| method), 81, 760, 768  | timed_dispense() (AFL.automation.loading.DigitalOutPressureControll  |
| test() (in module AFL.automation), 15, 819   | method), 236   |
|  | varifiledkdispense() (AFL.automation.loading.PressureController.Pressure   |

```
method), 45, 301, 302
                                                                                                                                                   tooltip (AFL.automation.prepare.PrepareWidget.Text
timed_dispense() (AFL.automation.loading.UltimusVPressureController
                        method), 375
                                                                                                                                                   tooltip (AFL.automation.prepare.PrepareWidget.VBox
timed_dispense() (AFL.automation.loading.UltimusVPressureController
                        method), 377
                                                                                                                                                   {\tt tooltip}\ (AFL. automation. prepare. Sample Series Widget. Button
to_dict()
                                    (AFL.automation.prepare.factory.Solution
                                                                                                                                                                            attribute), 572
                                                                                                                                                   tooltip (AFL.automation.prepare.SampleSeriesWidget.Checkbox
                        method), 715
to_dict() (AFL.automation.prepare.Solution method),
                                                                                                                                                                            attribute), 580
                                                                                                                                                   {\tt tooltip} (AFL. automation. prepare. Sample Series Widget. Float Text
to_stock_objects() (AFL.automation.prepare.StockBuilderWidgetastoithBuilderWidget_Model
                        method), 72, 649, 650
                                                                                                                                                   \verb+tooltip+ (AFL. automation. prepare. Sample Series Widget. HBox
toggleChannels() (AFL.automation.loading.MultiChannelRelay.MultiChata)ue3R0lay
                        method), 40, 262, 263
                                                                                                                                                   tooltip (AFL.automation.prepare.SampleSeriesWidget.IntText
toggleChannels() (AFL.automation.loading.SainSmartRelay.MultiGtwibuetRelay)
                                                                                                                                                   \verb|tooltip| (AFL. automation. prepare. Sample Series Widget. Label \\
                        method), 330
toggleChannels() (AFL.automation.loading.SainSmartRelay.SainSmattmitRetkay 615
                                                                                                                                                   tooltip(AFL.automation.prepare.SampleSeriesWidget.Text)
                        method), 51, 331, 332
toJSON() (AFL.automation.APIServer.Driver.PersistentConfig
                                                                                                                                                                            attribute), 636
                                                                                                                                                   {\tt tooltip} (AFL.automation.prepare.SampleSeriesWidget.VBox
                        method), 170
toJSON() (AFL.automation.shared.PersistentConfig.PersistentConfig attribute), 644
                        method), 83, 774, 776
                                                                                                                                                   tooltip(AFL.automation.prepare.SweepBuilderWidget.Button
token_in_blocklist_loader()
                                                                                                                                                                            attribute), 657
                        (AFL.automation.APIServer.APIServer.JWTManagooltip (AFL.automation.prepare.SweepBuilderWidget.Checkbox
                        method), 139
                                                                                                                                                                            attribute), 665
token_verification_failed_loader()
                                                                                                                                                   tooltip (AFL.automation.prepare.SweepBuilderWidget.HBox
                        (AFL.automation.APIServer.APIServer.JWTManager
                                                                                                                                                                            attribute), 673
                        method), 139
                                                                                                                                                   {\tt tooltip}\ (AFL. automation. prepare. Sweep Builder Widget. Label
token_verification_loader()
                                                                                                                                                                            attribute), 681
                        (AFL.automation.APIServer.APIServer.JWTManagooltip (AFL.automation.prepare.SweepBuilderWidget.Text
                        method), 139
                                                                                                                                                                            attribute), 701
tooltip(AFL. automation. prepare. Deck Builder Widget. Buttooltip(AFL. automation. prepare. Sweep Builder Widget. VBox
                        attribute), 406
                                                                                                                                                                            attribute), 709
tooltip (AFL.automation.prepare.DeckBuilderWidget.ChetchpvAFL.automation.prepare.DeckBuilderWidget.Layout
                        attribute), 414
                                                                                                                                                                            attribute), 449
tooltip (AFL.automation.prepare.DeckBuilderWidget.Dropdow(AFL.automation.prepare.PrepareWidget.Layout at-
                        attribute), 427
                                                                                                                                                                            tribute), 538
tooltip (AFL.automation.prepare.DeckBuilderWidget.HBoxop (AFL.automation.prepare.SampleSeriesWidget.Layout
                        attribute), 435
                                                                                                                                                                            attribute), 621
tooltip (AFL.automation.prepare.DeckBuilderWidget.Labatop (AFL.automation.prepare.SweepBuilderWidget.Layout
                        attribute), 443
                                                                                                                                                                            attribute), 687
tooltip (AFL.automation.prepare.DeckBuilderWidget.Texttouch() (AFL.automation.instrument.SeabreezeUVVis.Path
                        attribute), 461
                                                                                                                                                                            method), 212
tooltip (AFL.automation.prepare.DeckBuilderWidget.VBoxprint() (in module AFL.automation.shared.utilities),
                        attribute), 469
                                                                                                                                                                            88, 813, 814
{\tt tooltip} (AFL. automation. prepare. PrepareWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck BuilderWidget. Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Button \ {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Button \ {\tt
                        attribute), 499
                                                                                                                                                                            method), 406
tooltip(AFL.automation.prepare.PrepareWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.DeckBuilderWidget.Checkbotrait\_defaults()(AFL.automation.prepare.De
                        attribute), 506
                                                                                                                                                                            method), 414
{\tt tooltip} (AFL. automation. prepare. Prepare Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Builder Widget. Drop down {\tt rait\_defaults()} (AFL. automation. prepare. Deck {\tt Rait\_defaults()} (AFL. automation. prepare. Deck {\tt Rait\_defaults()} (AFL. automation. prepare. Deck {\tt Rait\_defaults()} (AFL. automation. prepar
                        attribute), 516
                                                                                                                                                                            method), 427
```

attribute), 524

attribute), 532

 ${\tt tooltip}~(AFL. automation. prepare. PrepareWidget. HBox ~~ {\tt trait\_defaults()}~(AFL. automation. prepare. DeckBuilderWidget. HBox ~~ {\tt trait\_defaults()}~(AFL. autom$ 

 ${\tt tooltip} \ (AFL. automation. prepare. Prepare Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Label \quad {\tt trait\_defaults()} \ (AFL. automation. prepare. Deck Builder Widget. Deck Builder Widget.$ 

*method*), 435

*method*), 443

- trait\_defaults() (AFL.automation.prepare.DeckBuilderWidget.Eugartis() (AFL.automation.prepare.DeckBuilderWidget.Button method), 453 class method), 406
- trait\_defaults() (AFL.automation.prepare.DeckBuilderWidget.Events() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 462 class method), 414
- trait\_defaults() (AFL.automation.prepare.DeckBuilderWialget\_Webents() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 469 class method), 428
- trait\_defaults() (AFL.automation.prepare.PrepareWidgataButtonvents() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 499 class method), 435
- trait\_defaults() (AFL.automation.prepare.PrepareWidgataGheckbents() (AFL.automation.prepare.DeckBuilderWidget.Label method), 506 class method), 444
- trait\_defaults() (AFL.automation.prepare.PrepareWidgatalitopekents() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 516 class method), 453
- trait\_defaults() (AFL.automation.prepare.PrepareWidgataHBoevents() (AFL.automation.prepare.DeckBuilderWidget.Text method), 524 class method), 462
- trait\_defaults() (AFL.automation.prepare.PrepareWidgendixtbedevents() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 532 class method), 469
- trait\_defaults() (AFL.automation.prepare.PrepareWidgetAidtyoewents() (AFL.automation.prepare.PrepareWidget.Button method), 542 class method), 499
- trait\_defaults() (AFL.automation.prepare.PrepareWidgetaFext\_events() (AFL.automation.prepare.PrepareWidget.Checkbox method), 556 class method), 507
- trait\_defaults() (AFL.automation.prepare.PrepareWidgata\int\_oevents() (AFL.automation.prepare.PrepareWidget.Dropdown method), 563 class method), 517
- trait\_defaults() (AFL.automation.prepare.SampleSeriet WädgeteWeintus() (AFL.automation.prepare.PrepareWidget.HBox method), 572 class method), 524
- trait\_defaults() (AFL.automation.prepare.SampleSerietWailgeteGhecksOx (AFL.automation.prepare.PrepareWidget.Label method), 580 class method), 533
- trait\_defaults() (AFL.automation.prepare.SampleSeriex WailgeteWeartEX) (AFL.automation.prepare.PrepareWidget.Layout method), 592 class method), 542
- trait\_defaults() (AFL.automation.prepare.SampleSerietWaidgeteWBnts() (AFL.automation.prepare.PrepareWidget.Text method), 599 class method), 556
- trait\_defaults() (AFL.automation.prepare.SampleSerietWailgeteWneTetxs() (AFL.automation.prepare.PrepareWidget.VBox method), 607 class method), 563
- method), 60/ class method), 563
  trait\_defaults() (AFL.automation.prepare.SampleSeriesWidget.Button
- method), 615

  class method), 573

  trait\_defaults() (AFL.automation.prepare.SampleSeries Wait yet Novus () (AFL.automation.prepare.SampleSeries Widget.Checkbox method), 625

  class method), 581
- trait\_defaults() (AFL.automation.prepare.SampleSerietWällteteWants() (AFL.automation.prepare.SampleSeriesWidget.FloatText
  method), 637

  class method), 592

  trait\_defaults() (AFL automation prepare SampleSerietWällteteWBnts() (AFL automation prepare SampleSerietWidget.HRoy
- trait\_defaults() (AFL.automation.prepare.SampleSerie\*\*WädgeteWBots() (AFL.automation.prepare.SampleSerie\*\*Widget.HBox method), 644 class method), 599
- trait\_defaults() (AFL.automation.prepare.SweepBuilder Wällger Battos() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 657 class method), 608
- trait\_defaults() (AFL.automation.prepare.SweepBuilder Weitligen Charles Ox (AFL.automation.prepare.SampleSeries Widget.Label method), 665 class method), 616
- trait\_defaults() (AFL.automation.prepare.SweepBuilderWällgerHebres() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 673 class method), 625
- trait\_defaults() (AFL.automation.prepare.SweepBuilder Wailger Lebtds() (AFL.automation.prepare.SampleSeriesWidget.Text method), 681 class method), 637
- trait\_defaults() (AFL.automation.prepare.SweepBuilder#Wältgenkarus() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 691 class method), 644
- trait\_defaults() (AFL.automation.prepare.SweepBuilderWidget.Button method), 702 class method), 657
- trait\_defaults() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 709 class method), 665

- trait\_events() (AFL.automation.prepare.SweepBuilderWindgirtHBas\_value() (AFL.automation.prepare.SampleSeriesWidget.Layou class method), 673 method), 626
- trait\_events() (AFL.automation.prepare.SweepBuilderWirdgintLhbel\_value() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 682 method), 687
- trait\_events() (AFL.automation.prepare.SweepBuilderWindgitLhasutvalue() (AFL.automation.prepare.SampleSeriesWidget.VBox class method), 691 method), 644
- trait\_events() (AFL.automation.prepare.SweepBuilderWidget.Butto class method), 702 method), 658
- trait\_events() (AFL.automation.prepare.SweepBuilderWidget.VBas\_value() (AFL.automation.prepare.SweepBuilderWidget.Checclass method), 709

  method), 666
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidgehBsttvalue() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 407 method), 673
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidgeh&evkBoue() (AFL.automation.prepare.SweepBuilderWidget.Labe method), 415 method), 682
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidgehBsopddwe() (AFL.automation.prepare.SweepBuilderWidget.Layo method), 428 method), 692
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidget.Has\_oxalue() (AFL.automation.prepare.SweepBuilderWidget.Text method), 435 method), 702
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 444 method), 709
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidget.Button method), 454 method), 407
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidgetEtxadata() (AFL.automation.prepare.DeckBuilderWidget.Checkbuilderwidget.Checkbuilderwid
- trait\_has\_value() (AFL.automation.prepare.DeckBuilderWidgem&Bodata() (AFL.automation.prepare.DeckBuilderWidget.Dropdo method), 469 method), 428
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgatiButmertadata() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 499 method), 436
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgatiChandchadata() (AFL.automation.prepare.DeckBuilderWidget.Label method), 507 method), 444
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgatiBroqueltachata() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 517 method), 454
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgatiftBmetadata() (AFL.automation.prepare.DeckBuilderWidget.Text method), 524 method), 462
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgetiltalmetadata() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 533 method), 470
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgetiltaymedtadata() (AFL.automation.prepare.PrepareWidget.Button method), 543 method), 499
- trait\_has\_value() (AFL.automation.prepare.PrepareWidget.Checkbox method), 556 method), 507
- trait\_has\_value() (AFL.automation.prepare.PrepareWidgativBmetadata() (AFL.automation.prepare.PrepareWidget.Dropdown method), 563 method), 517
- trait\_has\_value() (AFL.automation.prepare.SampleSerierWidgmeRaddata() (AFL.automation.prepare.PrepareWidget.HBox method), 573 method), 525
- trait\_has\_value() (AFL.automation.prepare.SampleSerierWidgeteCheckback) (AFL.automation.prepare.PrepareWidget.Label method), 581 method), 533
- trait\_has\_value() (AFL.automation.prepare.SampleSerierWittgmteFlodaFex() (AFL.automation.prepare.PrepareWidget.Layout method), 592 method), 543
- trait\_has\_value() (AFL.automation.prepare.SampleSerierWidgettBdata() (AFL.automation.prepare.PrepareWidget.Text method), 599 method), 556
- trait\_has\_value() (AFL.automation.prepare.SampleSerierWidgmeHualbata() (AFL.automation.prepare.PrepareWidget.VBox method), 608 method), 564
- trait\_has\_value() (AFL.automation.prepare.SampleSeriesWidgetAbdata() (AFL.automation.prepare.SampleSeriesWidget.Button method), 616 method), 573

- trait\_metadata() (AFL.automation.prepare.SampleSerietWaitger(GhestD):(AFL.automation.prepare.PrepareWidget.Label method), 581 method), 533
- trait\_metadata() (AFL.automation.prepare.SampleSeriexWaidgetWaidgetMatheuxText(AFL.automation.prepare.PrepareWidget.Layout method), 592 method), 543
- trait\_metadata() (AFL.automation.prepare.SampleSerietWaidgetWaidgetMares() (AFL.automation.prepare.PrepareWidget.Text method), 600 method), 556
- trait\_metadata() (AFL.automation.prepare.SampleSeriexWaidgetWaidgetWantEsx() (AFL.automation.prepare.PrepareWidget.VBox method), 564
- trait\_metadata() (AFL.automation.prepare.SampleSerietWailgetMandes!() (AFL.automation.prepare.SampleSeriesWidget.Button method), 616 method), 573
- trait\_metadata() (AFL.automation.prepare.SampleSeriexWaidgetNamesa() (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 581
- trait\_metadata() (AFL.automation.prepare.SampleSerietWaidgetNEmes() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 637 method), 592
- trait\_metadata() (AFL.automation.prepare.SampleSeriexWailgemMakes() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 645 method), 600
- trait\_metadata() (AFL.automation.prepare.SweepBuilder:WaitgenBmson() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 658 method), 608
- $\label{lem:condition} trait\_metadata() \ (AFL. automation. prepare. Sweep Builder \textit{Willge} \texttt{n} \textit{GiheckDoo} AFL. automation. prepare. Sample Series Widget. Label method), 666 \\ method), 616$
- trait\_metadata() (AFL.automation.prepare.SweepBuilderMilgenHibes() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 674 method), 626
- trait\_metadata() (AFL.automation.prepare.SweepBuilder\dilgen\dilg
- trait\_metadata() (AFL.automation.prepare.SweepBuilder Wäilgenhmess(t) (AFL.automation.prepare.SampleSeriesWidget.VBox method), 692 method), 645
- trait\_metadata() (AFL.automation.prepare.SweepBuilderWidget.Button method), 702 method), 658
- trait\_metadata() (AFL.automation.prepare.SweepBuilderWädgen\address() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 710

  method), 666
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgeaButmames() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 407 method), 674
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenChenkhnes() (AFL.automation.prepare.SweepBuilderWidget.Label method), 415 method), 682
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenDropdmes() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 428 method), 692
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenHB\_mames() (AFL.automation.prepare.SweepBuilderWidget.Text method), 436 method), 702
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenLubralames() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 444 method), 710
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenLtyonallues() (AFL.automation.prepare.DeckBuilderWidget.Button method), 454 method), 407
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenTextvalues() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 462 method), 415
- trait\_names() (AFL.automation.prepare.DeckBuilderWidgenYBaxalues() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 470 method), 428
- trait\_names() (AFL.automation.prepare.PrepareWidget.Brutairt\_values() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 499 method), 436
- trait\_names() (AFL.automation.prepare.PrepareWidget.Chrackbowalues() (AFL.automation.prepare.DeckBuilderWidget.Label method), 507 method), 444
- trait\_names() (AFL.automation.prepare.PrepareWidget.Panaidowalues() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 517 method), 454
- trait\_names() (AFL.automation.prepare.PrepareWidget.HBait\_values() (AFL.automation.prepare.DeckBuilderWidget.Text method), 525 method), 462

- trait\_values() (AFL.automation.prepare.DeckBuilderWidgait VB(); (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 470 method), 429
- trait\_values() (AFL.automation.prepare.PrepareWidgettHadious() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 499 method), 436
- trait\_values() (AFL.automation.prepare.PrepareWidgettCheicks()) (AFL.automation.prepare.DeckBuilderWidget.Label method), 507 method), 445
- trait\_values() (AFL.automation.prepare.PrepareWidgettIPaipts(Wn(AFL.automation.prepare.DeckBuilderWidget.Layout method), 517 method), 454
- trait\_values() (AFL.automation.prepare.PrepareWidget#Rixs() (AFL.automation.prepare.DeckBuilderWidget.Text method), 525 method), 463
- trait\_values() (AFL.automation.prepare.PrepareWidgetMadiets() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 533 method), 470
- trait\_values() (AFL.automation.prepare.PrepareWidget:Hungirus() (AFL.automation.prepare.PrepareWidget.Button method), 543 method), 500
- trait\_values() (AFL.automation.prepare.PrepareWidgetfFeaits() (AFL.automation.prepare.PrepareWidget.Checkbox method), 556 method), 508
- trait\_values() (AFL.automation.prepare.PrepareWidgett\dats() (AFL.automation.prepare.PrepareWidget.Dropdown method), 564 method), 518
- trait\_values() (AFL.automation.prepare.SampleSeriesWidgettBut)(nAFL.automation.prepare.PrepareWidget.HBox method), 573 method), 525
- trait\_values() (AFL.automation.prepare.SampleSeriesWidgettSkedAbEL.automation.prepare.PrepareWidget.Label method), 581 method), 534
- trait\_values() (AFL:automation.prepare.SampleSeriesWidgettEl@u(ReEL:automation.prepare.PrepareWidget.Layout method), 592 method), 543
- trait\_values() (AFL.automation.prepare.SampleSeriesWidgettIsB)x(AFL.automation.prepare.PrepareWidget.Text method), 600 method), 557
- trait\_values() (AFL.automation.prepare.SampleSeriesWidgattls(Te(AFL.automation.prepare.PrepareWidget.VBox method), 608 method), 564
- trait\_values() (AFL.automation.prepare.SampleSeriesWidget.Button method), 616 method), 574
- trait\_values() (AFL.automation.prepare.SampleSeriesWidgettExty) (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 582
- trait\_values() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 637 method), 593
- trait\_values() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 645 method), 600
- trait\_values() (AFL.automation.prepare.SweepBuilderWindgitBU) 6AFL.automation.prepare.SampleSeriesWidget.IntText method), 658 method), 609
- trait\_values() (AFL.automation.prepare.SweepBuilderWidgettElgetMcMatton.prepare.SampleSeriesWidget.Label method), 666 method), 617
- trait\_values() (AFL.automation.prepare.SweepBuilderWirkzitHBokAFL.automation.prepare.SampleSeriesWidget.Layout method), 674 method), 626
- trait\_values() (AFL.automation.prepare.SweepBuilderWindgirts(b)(AFL.automation.prepare.SampleSeriesWidget.Text method), 682 method), 638
- trait\_values() (AFL.automation.prepare.SweepBuilderWirdgittb@)dutFL.automation.prepare.SampleSeriesWidget.VBox method), 692 method), 645
- trait\_values() (AFL.automation.prepare.SweepBuilderWinkgitEQ) (AFL.automation.prepare.SweepBuilderWidget.Button method), 702 method), 658
- trait\_values() (AFL.automation.prepare.SweepBuilderWidgit\B)(AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 710 method), 666
- traits() (AFL.automation.prepare.DeckBuilderWidget.Buttoaits() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 407 method), 674
- traits() (AFL.automation.prepare.DeckBuilderWidget.Cherkbicus() (AFL.automation.prepare.SweepBuilderWidget.Label method), 415 method), 683

```
traits()(AFL.automation.prepare.SweepBuilderWidget.Layout
                                                                                                                                                                                                                                                                                                                                                                   58, 367, 373
                                                   method), 692
                                                                                                                                                                                                                                                                                                                type (AFL.automation.APIServer.APIServer.ServiceInfo
traits()(AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                                                                                                                                                                                                                                                                   attribute), 149
                                                  method), 703
                                                                                                                                                                                                                                                                                                                {\tt type}\, (AFL. automation. shared. Server Discovery. Async Service Info
traits()(AFL.automation.prepare.SweepBuilderWidget.VBox
                                                                                                                                                                                                                                                                                                                                                                   attribute), 781
                                                 method), 710
                                                                                                                                                                                                                                                                                                                type (AFL.automation.shared.ServerDiscovery.ServiceInfo
transfer() (AFL.automation.EpicsADLiveProcess.Client.Client
                                                                                                                                                                                                                                                                                                                                                                   attribute), 799
                                                  method), 27, 822, 823
                                                                                                                                                                                                                                                                                                                types (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
transfer() (AFL.automation.loading.OneSelectorBlowoutSampleCellt@baSelectiOrBlowoutSampleCell
                                                                                                                                                                                                                                                                                                               types (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                                                  method), 274
transfer() (AFL.automation.loading.OneSelectorBlowoutSampleCedtffibaSe)e7D4BlowoutSampleCell
                                                  method), 277
transfer() (AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell
                                                 method), 48, 311, 317
                                                                                                                                                                                                                                                                                                               UltimusVPressureController
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         in
transfer() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSampleCell.RsoXSolutionSamp
                                                  method), 50, 322, 328
                                                                                                                                                                                                                                                                                                                                                                    60, 376, 377
transfer() (AFL.automation.loading.TwoSelectorBlowoutSampleCell TwoSelectorBlowoutSampleCell
                                                 method), 59, 369, 374
                                                                                                                                                                                                                                                                                                                                                                   (AFL.automation.APIServer.APIServer.JWTManager
transfer() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy_OT2_Driver.Dummy
                                                  method), 67, 477, 479
                                                                                                                                                                                                                                                                                                               unlatch_shaker() (AFL.automation.prepare.Dummy_OT2_Driver.Dumm
transfer() (AFL.automation.prepare.OT2Client.OT2Client
                                                                                                                                                                                                                                                                                                                                                                   method), 66, 477, 479
                                                 method), 67, 484, 487
                                                                                                                                                                                                                                                                                                                unlink() (AFL.automation.instrument.SeabreezeUVVis.Path
transfer() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                                                                                                                                                                                                                                                                                   method), 212
                                                  method), 728
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.DeckBuilderWidget.Button
transform() (AFL.automation.shared.DataLabelerWidget.OrdinalEngodera), 407
                                                 method), 745
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
transform() (AFL.automation.shared.DiffractionLabeler.OrdinalEngodphod), 415
                                                 method), 766
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
transmit() (AFL.automation.APIServer.QueueDaemon.DataTrashcanethod), 429
                                                 method), 188
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.DeckBuilderWidget.HBox
trap_http_exception()
                                                                                                                                                                                                                                                                                                                                                                   method), 436
                                                  (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.DeckBuilderWidget.Label
                                                 method), 122
                                                                                                                                                                                                                                                                                                                                                                   method), 445
tubing (AFL. automation. loading. PushPull Selector Sample \textit{Gallo Bukirsse} () (AFL. automation. prepare. Deck Builder Widget. Layout tubing (AFL. automation. prepare. Deck Builder Widget. Deck Builder Widget.
                                                   attribute), 314
                                                                                                                                                                                                                                                                                                                                                                   method), 454
tubing (AFL. automation. loading. RSoXSS olution Sample Cell Tobiter ve() (AFL. automation. prepare. Deck Builder Widget. Text to the same part of the prepare of the pre
                                                 attribute), 325
                                                                                                                                                                                                                                                                                                                                                                    method), 463
tubing
                                                    (AFL.automation.loading.Tubing.Tubing
                                                                                                                                                                                                                                                                                                              unobserve() (AFL.automation.prepare.DeckBuilderWidget.VBox
                                                   tribute), 58, 362
                                                                                                                                                                                                                                                                                                                                                                    method), 470
tubing (AFL. automation. loading. Two Selector Blowout Sample Gall Twicks) (AFL. automation. prepare Widget. Button tubing (
                                                  attribute), 366
                                                                                                                                                                                                                                                                                                                                                                   method), 500
\textbf{Tubing} (class \ in AFL. automation. loading. PushPull Selector \textbf{SampleCeVO}) (AFL. automation. prepare. Prepare Widget. Checkbox and the property of the
                                                                                                                                                                                                                                                                                                                                                                   method), 508
\textbf{Tubing} (class \textit{ in AFL}. \textit{automation}. loading. \textit{RSoXSSolutionSampleSell} \textbf{ye} \textbf{()} (AFL. \textit{automation}. \textit{prepare}. \textit{PrepareWidget}. \textit{Dropdown} \textbf{()} \textbf{()}
                                                                                                                                                                                                                                                                                                                                                                   method), 518
Tubing (class in AFL.automation.loading.Tubing), 58, unobserve() (AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                                                                                                                                                                                                                                                   method), 525
\textbf{Tubing} \ (class\ in\ AFL. automation. loading. Two Selector Blow \textbf{\textit{qutSurveled}}) (AFL. automation. prepare. Prepare Widget. Label 1.1) (AFL. automation. prepare Widget. prepare Widget. prepare Widget. Prepare Widget. prepare Widget. prepa
                                                   366
                                                                                                                                                                                                                                                                                                                                                                   method), 534
TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                                          (class
                                                                                                                                                                                                                                                                                         in unobserve() (AFL.automation.prepare.PrepareWidget.Layout
                                                 AFL.automation.loading.OneSelectorBlowoutSampleCell), method), 543
                                                                                                                                                                                                                                                                                                                unobserve() (AFL.automation.prepare.PrepareWidget.Text
TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                          in
                                                                                                                                                                                                                                                                                                                                                                   method), 557
                                                 AFL.automation.loading.TwoSelectorBlowoutSampleCell).
```

```
unobserve() (AFL.automation.prepare.PrepareWidget.VBunobserve_all() (AFL.automation.prepare.PrepareWidget.Dropdown
        method), 564
                                                             method), 518
```

- unobserve() (AFL.automation.prepare.SampleSeriesWidgentblusserve\_all() (AFL.automation.prepare.PrepareWidget.HBox method), 526 method), 574
- unobserve() (AFL.automation.prepare.SampleSeriesWidgenCheedroe\_all() (AFL.automation.prepare.PrepareWidget.Label method), 582 method), 534
- unobserve() (AFL.automation.prepare.SampleSeriesWidgentolosserFewe\_all() (AFL.automation.prepare.PrepareWidget.Layout method), 544 *method*), 593
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetntBserve\_all() (AFL.automation.prepare.PrepareWidget.Text method), 557 method), 600
- unobserve() (AFL.automation.prepare.SampleSeriesWidgentNobEenre\_all() (AFL.automation.prepare.PrepareWidget.VBox method), 565 *method*), 609
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetnbdsdrve\_all() (AFL.automation.prepare.SampleSeriesWidget.Button method), 574
- unobserve() (AFL.automation.prepare.SampleSeriesWidgentlobsseurve\_all() (AFL.automation.prepare.SampleSeriesWidget.Checkbo method), 626 method), 582
- unobserve() (AFL.automation.prepare.SampleSeriesWidgentRbserve\_all() (AFL.automation.prepare.SampleSeriesWidget.FloatTe. *method*), 638 *method*), 593
- unobserve() (AFL.automation.prepare.SampleSeriesWidgent\( \text{Bae}\)rve\_all() (AFL.automation.prepare.SampleSeriesWidget.HBox *method*), 645 method), 601
- unobserve() (AFL.automation.prepare.SweepBuilderWidgunBhserve\_all() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 609 *method*), 658
- unobserve() (AFL.automation.prepare.SweepBuilderWidgan6bsekbre\_all() (AFL.automation.prepare.SampleSeriesWidget.Label *method*), 666 method), 617
- unobserve() (AFL.automation.prepare.SweepBuilderWidganblbserve\_all() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 674 method), 627
- unobserve() (AFL.automation.prepare.SweepBuilderWidgenbbserve\_all() (AFL.automation.prepare.SampleSeriesWidget.Text method), 683 method), 638
- unobserve() (AFL.automation.prepare.SweepBuilderWidganbbsonve\_all() (AFL.automation.prepare.SampleSeriesWidget.VBox *method*), 692 method), 646
- unobserve() (AFL.automation.prepare.SweepBuilderWidgetn&bserve\_all() (AFL.automation.prepare.SweepBuilderWidget.Button *method*), 703 method), 659
- unobserve() (AFL.automation.prepare.SweepBuilderWidgentVBserve\_all() (AFL.automation.prepare.SweepBuilderWidget.Checkbo method), 710 method), 667
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWinderBrutenall() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 675 method), 408
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWindgesetheeklackd () (AFL.automation.prepare.SweepBuilderWidget.Label method), 416 method), 683
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWindgetsDrvp.davlk() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 693 *method*), 429
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWindgetsHBree\_all() (AFL.automation.prepare.SweepBuilderWidget.Text *method*), 437 method), 703
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWindersEurbel\_all() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 711 method), 445
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWindgickdingError, 811

*method*), 617

- unqueued() (AFL.automation.APIServer.Driver.Driver method), 455
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWidget.Texmethod), 21, 167, 171
  - unqueued() (AFL.automation.APIServer.DummyDriver.Driver *method*), 463
- unobserve\_all() (AFL.automation.prepare.DeckBuilderWidget.VBonethod), 174
  - method), 471 unqueued() (AFL.automation.APIServer.DummyDriver.DummyDriver
- unobserve\_all() (AFL.automation.prepare.PrepareWidget.Button method), 179
  - unqueued() (AFL.automation.APIServer.DummyOT2Driver.Driver *method*), 500
- unobserve\_all() (AFL.automation.prepare.PrepareWidget.Checkbonethod), 181
  - unqueued() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver method), 508

| method), 185   | method), 725  |
|--|---|
| unqueued() (AFL.automation.instrument.DummySAS.Drivænqueue   |   |
| method), 194   | method), 732  |
| unqueued() (AFL.automation.instrument.DummySAS.DummyQAdSue   |   |
| method), 198   | method), 736  |
| unqueued() (AFL.automation.instrument.I22SAXS.Driver unqueue   |   |
| method), 200   | method), 20, 161, 164   |
| unqueued() (AFL.automation.instrument.I22SAXS.I22SAX\( \) inqueue  |   |
| method), 204   | method), 246  |
| unqueued() (AFL.automation.instrument.SeabreezeUVVis.Darqueeue   |   |
| method), 206   | method), 418  |
| unqueued() (AFL.automation.instrument.SeabreezeUVVis.Singlueue   |   |
| method), 218   | method), 482  |
| unqueued() (AFL.automation.instrument.SpecScreen_DrivanDanae   |   |
| method), 220   | method), 486  |
| unqueued() (AFL.automation.instrument.SpecScreen_DrivanspecSe  |   |
|  | method), 584  |
| method), 223   |   |
| unqueued() (AFL.automation.loading.LoadStopperDriver.Dniqueue  |   |
| method), 247   | method), 723  |
| unqueued() (AFL.automation.loading.LoadStopperDriver.LunartStrap   |   |
| method), 252   | method), 730  |
| unqueued() (AFL.automation.loading.OneSelectorBlowoutSnrqqie6  |   |
| method), 268   | (AFL.automation.APIServer.APIServer.Zeroconf                                      |
| unqueued() (AFL.automation.loading.OneSelectorBlowoutSampleC   | · ·   |
|  | ter_all_services()  |
| unqueued() (AFL.automation.loading.OneSelectorBlowoutSampleC   |   |
| method), 278   | method), 807  |
| unqueued() (AFL.automation.loading.PneumaticPressureSumplgCs   |   |
| method), 282   | (AFL.automation.APIServer.APIServer.Zeroconf                                      |
| $unqueued () \ (AFL. automation. loading. Pneumatic Pressure Sample Cells and Cells a$ |   |
|  | ter_service()   |
| $unqueued () \ (AFL. automation. loading. Pneumatic Sample Cell. Driver and the properties of the pro$ |   |
| method), 292   | method), 807  |
| unqueued() (AFL.automation.loading.PneumaticSampleCelhBetwj  |   |
| method), 297   | AFL.automation.APIServer.APIServer), 96   |
| unqueued() (AFL.automation.loading.PushPullSelectorSamplaGed)  | DrAEL.automation.APIServer.Driver.PersistentConfig                                |
| method), 307   | method), 171  |
| $unqueued () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. loading. PushPull Selector Sample College () \ (AFL. automation. loading. loading. PushPull Selector Sample College () \ (AFL. automation. loading. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. PushPull Selector Sample College () \ (AFL. automation. loading. loading. loading. loading. loading. PushPull Selector Sample College () \ (AFL. automation.$ | PuAlFRudistederattivstaArpls&eler.QueueDaemon.DataTrashcan                        |
| method), 313   | method), 189  |
| $unqueued () \ (AFL. automation. loading. RSoXSS olution Sample Call. Description of the control of the contro$ | $\dot{\mathcal{P}}$ iv(ArFL.automation.loading.OneSelectorBlowoutSampleCell.defau |
| method), 318   | method), 280  |
| unqueued() (AFL.automation.loading.RSoXSSolutionSampleCall.eX  | So <b>(ASSblattitomSattiepldGæll</b> ing.PneumaticPressureSampleCell.default      |
| method), 324   | method), 290  |
| unqueued() (AFL.automation.loading.TwoSelectorBlowoutSpdateC   | (AFA) Nantomation.loading.PneumaticSampleCell.defaultdict                         |
| method), 364   | method), 300  |
| unqueued() (AFL.automation.loading.TwoSelectorBlowoutSpdate()  | <b>Əl(AFA) SaltamatBbayloatSagaPheGAH</b> ullSelectorSampleCell.defaultdi         |
| method), 371   | method), 316  |
| unqueued() (AFL.automation.prepare.Dummy_OT2_DriveupDate()   |   |
| method), 473   | method), 327  |
| unqueued() (AFL.automation.prepare.Dummy_OT2_Driveupliane)   |   |
| method), 478   | method), 373  |
| unqueued() (AFL.automation.sample.CastingServer.CastingSeates(   |   |
| method), 721   | method), 756  |
| unqueued() (AFL.automation.sample.CastingServer.Driverupdate(  |   |
|  | ,   |

```
method), 79, 751, 757
                                        method), 772
update() (AFL.automation.shared.PersistentConfig.Persistent@protocol_order_preview_cb()
                                        method), 83, 775, 776
                                                                                                                                                                                                                                                                                              (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget
update_colors() (AFL.automation.shared.DatasetWidget.DatasetWidgetbod), 546
                                        method), 79, 751, 757
                                                                                                                                                                                                                                                     update_protocol_order_preview_cb()
update_colorscale()
                                                                                                                                                                                                                                                                                              (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
                                        (AFL.automation.shared.DatasetWidget_DatasetWidget_Viewnethod), 70, 628, 646
                                        method), 80, 754, 758
                                                                                                                                                                                                                                                     update_record() (AFL.automation.APIServer.APIServer.ServiceInfo
update_component_row_cb()
                                                                                                                                                                                                                                                                                              method), 152
                                        (AFL.automation.prepare.PrepareWidget.SweepBuibdarWeidgetord() (AFL.automation.shared.ServerDiscovery.AsyncService
                                        method), 549
                                                                                                                                                                                                                                                                                              method), 779
                                                                                                                                                                                                                                                     \verb"update_record"() (AFL. automation. shared. Server Discovery. A sync Service and Server Discovery. A sync Discovery. A sync Server Discovery. A sync Discove
update_component_row_cb()
                                        (AFL.automation.prepare.SweepBuilderWidget.SweepBuildenWildget), 784
                                        method), 73, 694, 711
                                                                                                                                                                                                                                                     update_record() (AFL.automation.shared.ServerDiscovery.ServiceBrown
update_composition_colors()
                                                                                                                                                                                                                                                                                              method), 796
                                        (AFL.automation.shared.DataLabelerWidget.DataIpdarlerWieword() (AFL.automation.shared.ServerDiscovery.ServiceInfo
                                        method), 78, 739, 748
                                                                                                                                                                                                                                                                                              method), 802
update_composition_plot()
                                                                                                                                                                                                                                                     update_sample_dim()
                                        (AFL. automation. shared. Dataset Widget. Dataset Widget
                                                                                                                                                                                                                                                                                              (AFL.automation.shared.DatasetWidget.DatasetWidget
                                        method), 79, 751, 757
                                                                                                                                                                                                                                                                                              method), 79, 752, 757
update_deck_graphic_cb()
                                                                                                                                                                                                                                                     update_scattering_plot()
                                        (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWAfgetautomation.shared.DatasetWidget.DatasetWidget
                                                                                                                                                                                                                                                                                              method), 79, 751, 757
                                        method), 65, 420, 471
update_deck_graphic_cb()
                                                                                                                                                                                                                                                     update_selected() (AFL.automation.shared.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.DatasetWidget.Dat
                                        (AFL.automation.prepare.PrepareWidget.DeckBuilderWidgenethod), 80, 754, 758
                                        method), 509
                                                                                                                                                                                                                                                     update_service() (AFL.automation.APIServer.APIServer.Zeroconf
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget155
                                        method), 79, 752, 757
                                                                                                                                                                                                                                                     \verb"update_service" () (AFL. automation. shared. Server Discovery. Async Zerocomunication and the property of 
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget_T\sqrt{lew}
                                                                                                                                                                                                                                                     \verb"update_service" () (AFL. automation. shared. Server Discovery. Zero configuration of the property of the p
                                        method), 80, 754, 758
update_extract_coords()
                                                                                                                                                                                                                                                                                              method), 806
                                        (AFL.automation.shared.DatasetWidget.DatasetWidgetate_sort_order_cb()
                                                                                                                                                                                                                                                                                              (AFL. automation. prepare. Prepare Widget. Sample Series Widget
                                        method), 79, 752, 757
update_location_check_cb()
                                                                                                                                                                                                                                                                                              method), 547
                                        (AFL.automation.prepare.PrepareWidget.StockBuilpdaWidgeort_order_cb()
                                        method), 548
                                                                                                                                                                                                                                                                                              (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSeriesWidget.SampleSe
update_location_check_cb()
                                                                                                                                                                                                                                                                                              method), 71, 629, 646
                                        (AFL.automation.prepare.StockBuilderWidget.StockBuilder\(\text{StockBuilder}\)\(\text{StockBuilder}\)\(\text{AFL.automation.loading.LoadStopperDriver.StopLoad\)
                                        method), 72, 648, 650
                                                                                                                                                                                                                                                                                              method), 258
update_mixing_well_preview_cb()
                                                                                                                                                                                                                                                     update_status() (AFL.automation.loading.LoadStopperDriver.StopLoad
                                        (AFL.automation.prepare.PrepareWidget.SampleSeriesWidgethod), 261
                                        method), 547
                                                                                                                                                                                                                                                     update_status() (AFL.automation.loading.SensorCallbackThread.Sensor
                                                                                                                                                                                                                                                                                              method), 54, 343, 353
update_mixing_well_preview_cb()
                                        (AFL.automation.prepare.SampleSeriesWidget.SampdeSee_iesVeitIges() (AFL.automation.loading.SensorCallbackThread.Simple
                                        method), 71, 629, 646
                                                                                                                                                                                                                                                                                              method), 347
update_plot() (AFL.automation.shared.DataLabelerWidgmadateLsbakers*(@m(AFL.automation.loading.SensorCallbackThread.StopL
                                        method), 77, 739, 748
                                                                                                                                                                                                                                                                                              method), 350
update_plot() (AFL.automation.shared.DataLabelerWidgendDateLsbateusWidleAFL.automation.loading.SensorCallbackThread.StopL
                                        method), 77, 740, 748
                                                                                                                                                                                                                                                                                              method), 353
update_plot() (AFL.automation.shared.DiffractionLabelapldiffractionLubelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLabelapldiffractionLab
                                        method), 81, 760, 768
                                                                                                                                                                                                                                                                                              method), 89, 720, 730
update_plot() (AFL.automation.shared.DiffractionLabeleupdiffractionphladerVieutext()
                                        method), 81, 762, 769
                                                                                                                                                                                                                                                                                               (AFL.automation.APIServer.APIServer.Flask
update_plots() (AFL.automation.shared.DatasetWidget.DatasetWidgethod), 116
```

| update_ternary_colors() (AFL.automation.shared.DiffractionLabeler.Diff |  |
|--|--|
| method), 82, 762, 769  | validate_sweep_cb()  |
|  | eStateChangAFL.automation.prepare.PrepareWidget.SweepBuilderWidget   |
| attribute), 802  | method), 549   |
| url_build_error_handlers   | <pre>validate_sweep_cb()</pre>   |
| (AFL.automation.APIServer.APIServer.Flask                              | (AFL. automation. prepare. Sweep Builder Widget. Sweep Builder W   |
| attribute), 114  | method), 73, 694, 711  |
| url_default_functions  | value (AFL.automation.instrument.SeabreezeUVVis.Eq   |
| (AFL.automation.APIServer.APIServer.Flask                              | attribute), 208  |
| attribute), 132  | ${\tt value}  (AFL. automation. prepare. Deck Builder Widget. Check box$   |
| url_defaults() (AFL.automation.APIServer.APIServer.                    |  |
| method), 130   | ${\tt value}(AFL. automation. prepare. Deck Builder Widget. Drop down$   |
| <pre>url_for() (AFL.automation.APIServer.APIServer.Flask</pre>         | attribute), 429  |
| method), 124   | value (AFL.automation.prepare.DeckBuilderWidget.Label  |
| url_map (AFL.automation.APIServer.APIServer.Flask                      | attribute), 445  |
| attribute), 114  | value (AFL.automation.prepare.DeckBuilderWidget.Text   |
| url_map_class(AFL.automation.APIServer.APIServer.F                     |  |
| attribute), 113  | value (AFL.automation.prepare.PrepareWidget.Checkbox   |
| url_rule_class(AFL.automation.APIServer.APIServer.                     |  |
| attribute), 113  | value (AFL.automation.prepare.PrepareWidget.Dropdown   |
| url_value_preprocessor()   | attribute), 518  |
| (AFL.automation.APIServer.APIServer.Flask                              | value (AFL.automation.prepare.PrepareWidget.Label  |
| method), 130   | attribute), 534  |
| url_value_preprocessors  | value (AFL.automation.prepare.PrepareWidget.Text at-   |
| (AFL.automation.APIServer.APIServer.Flask                              | tribute), 557  |
| attribute), 131  | value (AFL.automation.prepare.SampleSeriesWidget.Checkbox  |
| usbrelay (in module AFL.automation.loading.SainSmartF                  |  |
|  |  |
| 51, 329, 332   | value (AFL.automation.prepare.SampleSeriesWidget.FloatText   |
| use_x_sendfile (AFL.automation.APIServer.APIServer.                    |  |
| property), 112   | value (AFL.automation.prepare.SampleSeriesWidget.IntText   |
| user_identity_loader()   | attribute), 609  |
|  | agedue (AFL.automation.prepare.SampleSeriesWidget.Label  |
| method), 140   | attribute), 617  |
| user_lookup_error_loader()   | value (AFL.automation.prepare.SampleSeriesWidget.Text  |
| (AFL.automation.APIServer.APIServer.JWTMan                             |  |
| method), 140   | ${\tt value} \ (AFL. automation. prepare. Sweep Builder Widget. Checkbox$  |
| user_lookup_loader()   | attribute), 667  |
| (AFL.automation.APIServer.APIServer.JWTMan                             | aagadue (AFL.automation.prepare.SweepBuilderWidget.Label   |
| method), 140   | attribute), 683  |
| 1.7  | ${\tt value}(AFL. automation. prepare. Sweep Builder Widget. Text$   |
| V  | attribute), 703  |
| V40nly (AFL.automation.APIServer.APIServer.IPVersion attribute), 135   | values() (AFL.automation.APIServer.Driver.PersistentConfig method), 171  |
| V40nly (AFL automation shared Server Discovery IPVersia                | on alues() (AFL.automation.APIServer.QueueDaemon.DataTrashcan  |
| attribute), 788  | method), 189   |
| V60nly (AFL.automation.APIServer.APIServer.IPVersion                   | values() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defau  |
| attribute), 135  | method), 280   |
| V60nly (AFI automation shared Samuer Discovery IDVanci                 | $_{OH}$ values() (AFL.automation.loading.PneumaticPressureSampleCell.default   |
|  | method), 290   |
| attribute), 788  | values() (AFL.automation.loading.PneumaticSampleCell.defaultdict   |
| validate_sample_series()   | method), 300   |
| (AFL.automation.prepare.Deck method), 384                              | values() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdi  |
| JUT  | , and the second |

method), 316

| values() (AFL.automation.loading.RSoXSSolutionSample \delta l.defaultdict   |
|---|
| method), 327 (AFI automation APIServer Client Client  |
| values() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultidicto, 161, 164   |
| method), 373 wait() (AFL.automation.loading.LoadStopperDriver.Client values() (AFL.automation.shared.DatasetWidget.defaultdict method), 246           |
| 1 D 756   |
| method), 756  wait() (AFL.automation.prepare.DeckBuilderWidget.Client values() (AFL.automation.shared.PersistentConfig.MutableMappingmethod), 418     |
| method), 772 wait () (AFL automation prepare OT2 Client Client  |
| values() (AFL.automation.shared.PersistentConfig.PersistentConfig method), 482  |
| method), 775  wait() (AFL.automation.prepare.OT2Client.OT2Client  WBox (class in AFL system ation prepare DeckBuilderWidert)                          |
| VBox (class in AFL.automation.prepare.DeckBuilderWidget), method), 486  |
| VBox (class in AFL.automation.prepare.PrepareWidget), wait() (AFL.automation.prepare.SampleSeriesWidget.Client method), 584                           |
| 557 wait() (AFL.automation.sample.CastingServer.Client  |
| VBox (class in AFL.automation.prepare.SampleSeriesWidget), method), 723   |
| 638 wait() (AFL.automation.sample.CastingServer.OT2Client   |
| VBox (class in AFL.automation.prepare.SweepBuilderWidget), method), 730   |
| Walt_dosage_fillsfied()   |
| AFL.automation.loading.DoubleViciMultiposSelector), (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePmethod), 35, 231, 233                     |
| 238 webapp() (AFL.automation.APIServer.APIServer  |
| ViciMultiposSelector (class in method), 18, 100, 158  |
| AFL. $automation. loading. ViciMultipos Selector), weight (AFL. automation. APIS erver. APIS erver. ServiceInfo$                                      |
| 61, 379, 380 attribute), 149  |
| view_functions (AFL.automation.APIServer.APIServer.Flaskght (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo attribute), 131                   |
| attribute), 131 visibility (AFL.automation.prepare.DeckBuilderWidget.Laxout (AFL.automation.shared.ServerDiscovery.ServiceInfo                        |
| attribute), 449   |
| visibility (AFL.automation.prepare.PrepareWidget.Layout dget_types (AFL.automation.prepare.DeckBuilderWidget.Button                                   |
| attribute), 556 attribute   |
| visibility (AFL.automation.prepare.SampleSeriesWidget, Layout_types (AFL.automation.prepare.DeckBuilderWidget.Checkbox                                |
| attribute), 021   |
| visibility (AFL.automation.prepare.SweepBuilderWidget_Lagett_types (AFL.automation.prepare.DeckBuilderWidget.Dropdown attribute), 687 attribute), 429 |
| volume (AFL.automation.prepare.Component.Component widget_types (AFL.automation.prepare.DeckBuilderWidget.HBox  |
| property), 64, 396, 398 attribute), 437   |
| volume (AFL.automation.prepare.factory.Solution prop-widget_types (AFL.automation.prepare.DeckBuilderWidget.Label                                     |
| erty), 715  volume (AEL sutemetion propers Solute property) 380   |
| volume (AFL.automation.prepare.Solute property), 389 volume (AFL.automation.prepare.Solution property), attribute) 455                                |
| volume (AFL.automation.prepare.Solution property), attribute), 455  392 widget_types (AFL.automation.prepare.DeckBuilderWidget.Text                   |
| volume (AFL.automation.prepare.Solvent property), 394 attribute) 463  |
| volume() (AFL.automation.loading.PushPullSelectorSampleCellTubinges (AFL.automation.prepare.DeckBuilderWidget.VBox                                    |
| memoa, 514 $attailanta$ , 471   |
| volume() (AFL.automation.loading.RSoXSSolutionSampleCellTubing) (AFL.automation.prepare.PrepareWidget.Button  |
| attribute), 500   |
| volume() (AFL.automation.loading.lubing widget_types (AFL.automation.prepare.PrepareWidget.Checkbox method), 58, 362 attribute), 508                  |
| volume() (AFL.automation.loading.TwoSelectorBlowoutSampleCell_Typesg(AFL.automation.prepare.PrepareWidget.Dropdown                                    |
| method), 300 attribute) 518   |
| volume_fraction(AFL.automation.prepare.factory.Solution_types(AFL.automation.prepare.PrepareWidget.HBox   |
| property), 713 attribute), 526  |
| volume_fraction (AFL.automation.prepare.Solution property), 392 widget_types (AFL.automation.prepare.PrepareWidget.Label attribute) 534               |
| property), 392 attribute), 534  |

- widget\_types (AFL.automation.prepare.PrepareWidget.Lowindgets (AFL.automation.prepare.PrepareWidget.Button attribute), 544 attribute), 540
- widget\_types (AFL.automation.prepare.PrepareWidget.Texidgets (AFL.automation.prepare.PrepareWidget.Checkbox attribute), 557 attribute), 508
- widget\_types (AFL.automation.prepare.PrepareWidget.Viadgets (AFL.automation.prepare.PrepareWidget.Dropdown attribute), 565 attribute), 518
- widget\_types (AFL.automation.prepare.SampleSeriesWidgettsn(AFL.automation.prepare.PrepareWidget.HBox attribute), 574 attribute), 526
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetdGktdsk&AFL.automation.prepare.PrepareWidget.Label attribute), 582 attribute), 584
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetdSlettsTleAFL.automation.prepare.PrepareWidget.Layout attribute), 593 attribute), 544
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetdgets (AFL.automation.prepare.PrepareWidget.Text attribute), 601 attribute), 557
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetClayetText(AFL.automation.prepare.PrepareWidget.VBox attribute), 609 attribute), 565
- widget\_types (AFL.automation.prepare.SampleSeriesWidgettes (AFL.automation.prepare.SampleSeriesWidget.Button attribute), 617 attribute), 574
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetdlgestest(AFL.automation.prepare.SampleSeriesWidget.Checkbox attribute), 627 attribute), 582
- widget\_types (AFL.automation.prepare.SampleSeriesWidgetEs(AFL.automation.prepare.SampleSeriesWidget.FloatText attribute), 638 attribute), 593
- widget\_types (AFL.automation.prepare.SampleSeriesWidgettBox (AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 646 attribute), 601
- widget\_types (AFL.automation.prepare.SweepBuilderWidgettBetton(AFL.automation.prepare.SampleSeriesWidget.IntText attribute), 659 attribute), 659
- widget\_types (AFL.automation.prepare.SweepBuilderWidgitlGhtsklAtkL.automation.prepare.SampleSeriesWidget.Label attribute), 667 attribute), 617
- widget\_types (AFL.automation.prepare.SweepBuilderWidgettBBcs:(AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 675 attribute), 627
- widget\_types (AFL.automation.prepare.SweepBuilderWidgetdbekel(AFL.automation.prepare.SampleSeriesWidget.Text attribute), 683 attribute), 688
- widget\_types (AFL.automation.prepare.SweepBuilderWidgettlbetrautAFL.automation.prepare.SampleSeriesWidget.VBox attribute), 693 attribute), 646
- widget\_types (AFL.automation.prepare.SweepBuilderWidgettBetts (AFL.automation.prepare.SweepBuilderWidget.Button attribute), 703 attribute), 659
- widget\_types (AFL.automation.prepare.SweepBuilderWidgetdyBos (AFL.automation.prepare.SweepBuilderWidget.Checkbox attribute), 711 attribute), 667
- $\label{eq:continuity} \begin{tabular}{ll} widgets (AFL. automation. prepare. Sweep Builder Widget. But windgets (AFL. automation. prepare. Sweep Builder Widget. HBox attribute), 408 \\ attribute), 675 \\ \end{tabular}$
- widgets (AFL.automation.prepare.DeckBuilderWidget.Chewklidgets (AFL.automation.prepare.SweepBuilderWidget.Label attribute), 416 attribute), 683
- widgets (AFL.automation.prepare.DeckBuilderWidget.Drowddgets (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 429 attribute), 693
- widgets (AFL.automation.prepare.DeckBuilderWidget.HBoxidgets (AFL.automation.prepare.SweepBuilderWidget.Text attribute), 437 attribute), 703
- widgets (AFL.automation.prepare.DeckBuilderWidget.Labwidgets (AFL.automation.prepare.SweepBuilderWidget.VBox attribute), 445 attribute), 711
- widgets (AFL.automation.prepare.DeckBuilderWidget.Laywirdth (AFL.automation.prepare.DeckBuilderWidget.Layout attribute), 455 attribute), 449
- widgets (AFL.automation.prepare.DeckBuilderWidget.Textwidth (AFL.automation.prepare.PrepareWidget.Layout attribute), 463 attribute), 538
- widgets (AFL.automation.prepare.DeckBuilderWidget.VBowidth (AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 471 attribute), 621

```
width (AFL.automation.prepare.SweepBuilderWidget.Layout
        attribute), 687
with_name() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 214
with_stem() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 214
with_suffix()(AFL.automation.instrument.SeabreezeUVVis.Path
        method), 214
withdraw() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
        method), 35, 231, 233
withdraw() (AFL. automation. loading. Chemyx Syringe Pump. Syringe Pump.
        method), 232
withdraw() (AFL.automation.loading.DummyPump.DummyPump
        method), 38, 241, 243
withdraw() (AFL.automation.loading.DummyPump.SyringePump
        method), 242
withdraw() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePump
        method), 41, 265, 267
withdraw() (AFL.automation.loading.NE1kSyringePump.SyringePump
        method), 266
withdraw() (AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump
        method), 46, 303, 305
withdraw() (AFL.automation.loading.PressureControllerAsPump.SyringePump
        method), 305
withdraw() (AFL.automation.loading.SyringePump.SyringePump
        method), 58, 361
write()
              (AFL.automation.prepare.ComponentDB
        method), 382
write_bytes() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 212
write_text() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 212
wsgi_app() (AFL.automation.APIServer.APIServer.Flask
        method), 134
X
xarray_to_bytes()
                                             module
                               (in
        AFL.automation.shared.utilities), 88, 814
Ζ
zc (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
        attribute), 779
zc (AFL.automation.shared.ServerDiscovery.ServiceBrowser
        attribute), 794
Zeroconf (class in AFL.automation.APIServer.APIServer),
Zeroconf (class in AFL.automation.shared.ServerDiscovery),
         803
```