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	11
6	Reference	13
7	Module Documentation	789
8	Indices and tables	797
Рy	ython Module Index	799
In	dex	801

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-scatteri	epics, sans, mantid	Support for neutron scattering instrumentation
nice-neutron-sca	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.

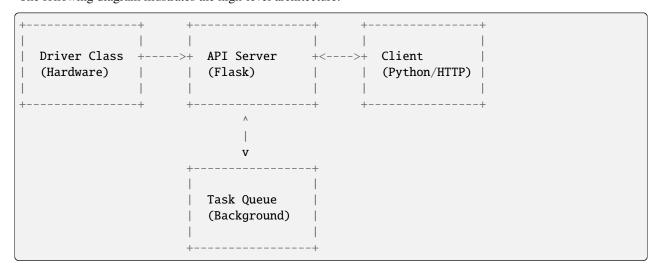
5.1.1 Core Concepts

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

AFL.automation.sample_env

AFL.automation.shared

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.
<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.

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

```
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()
```

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

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

```
 \begin{array}{c} ceil(x,/) & \text{Return the ceiling of x as an Integral.} \\ listify(obj) & \\ makeRegistrar() & \\ sqrt(x,/) & \text{Return the square root of x.} \\ \end{array}
```

Classes

unqueued()

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

PersistentConfig(path[, defaults, ...])

A dictionary-like class that serializes changes to disk
```

```
AFL.automation.APIServer.Driver.makeRegistrar()
```

class AFL.automation.APIServer.Driver.Driver(name, defaults=None, overrides=None)

```
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

	he ceiling of x as an Integral.
listify(obj)	
sqrt(x, /) Return the	he 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, l)	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.
<pre>QueueDaemon(app, driver, task_queue, history)</pre>	

```
__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, ...])
```

```
__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

```
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, ...[, ...])
```

__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)
```

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.

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	vided by Chelliya.
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
LoadStopperDriver	
MultiChannelRelay	
NE1kSyringePump	
OneSelectorBlowoutSampleCell	
PneumaticPressureSampleCell	
PneumaticSampleCell	
PressureController	
PressureControllerAsPump	
PushPullSelectorSampleCell	
RSoXSSolutionSampleCell	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
SyringePump	
Tubing	
TwoSelectorBlowoutSampleCell	
UltimusVPressureController	
ViciMultiposSelector	
30	Chapter 6. Reference

AFL.automation.loading.CetoniMultiPosValve

Classes

```
CetoniMultiPosValve(parentpump[, portlabels])
FlowSelector()
```

```
__init__(parentpump, portlabels={})
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)

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)
     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

Classes

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

```
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
                    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.
     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

Classes

```
rlowSelector()

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

AFL.automation.loading.LoadStopperDriver

```
Client([ip, port, username, interactive])
                                                   Communicate with APIServer
 Driver(name[, defaults, overrides])
 LoadStopperDriver(sensor[, load_client, ...])
                                                   Driver for stopping loads
 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': 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()
     reset_stopper()
```

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()
```

```
__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 separate 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

 $\textbf{class} \ AFL. automation. loading. One Selector Blowout Sample Cell. \textbf{One Selector Blowout Sample Cell} (\textit{pump}, \textit{pump}, \textit$

```
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

AFL.automation.loading.PneumaticPressureSampleCell

Classes

<pre>Driver(name[, defaults, overrides])</pre>	
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,

relayboard,
digitalin=None,
rinse1_tank_level=950,
rinse2_tank_level=950,
waste_tank_level=0,
load_stopper=None,
robot_interlock_host=N
overrides=None)

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

Driven by a variable-pressure regulator.

```
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

rinseAll()

setRinseLevel(vol)

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

      PneumaticSampleCell(pump, relayboard[, ...])
      Class for a sample cell consisting of a push-through, pneumatically-closed sample loader.

      SampleCell()
      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')
```

```
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

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

```
PressureControllerAsPump(pressure_controller)

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

SyringePump()
```

```
class AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump(pressure_controller,
                                                                                              dis-
                                                                                              pense pressure=3.5,
                                                                                              im-
                                                                                              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()
```

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

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

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

```
selec-
tor,
ncells=1,
thick-
ness=None,
catch to sel vol=None,
cell_to_sel_vol=None,
sv-
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
     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

<pre>Driver(name[, defaults, overrides])</pre>	
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

```
\textbf{class} \ \texttt{AFL}. automation. loading. RSoXSSolution Sample Cell. \textbf{RSoXSSolution Sample Cell} (\textit{pump}, \textit{selector}, \textit{pump}, \textit
```

```
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
```

Classes

```
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':FE05000000000FD\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':FE050007F00F7\r\n'], [b':FE0500080000F5\r\n', b':FE050008FF00F6\r\n'], [b':FE050000FF00F7\r\n'], [b':FE050009FF00F5\r\n'], [b':FE05000A0000F3\r\n', b':FE05000AFF00F4\r\n'], [b':FE05000B0000F2\r\n', b':FE05000D0000F0\r\n'], [b':FE05000DFF00F1\r\n', b':FE05000E0000FF\r\n'], [b':FE05000EFF00F5\r\n'], [b':FE05000EFF00F5\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n']]
```

"" Sainsmart 16-Channel 9-36v USB Relay Module (CH341 chip)(sku# 101-70-208) 1. Requires CH341 Windows driver installed (see http://wiki.sainsmart.com/index.php/101-70-208) 2. The hex table provided by Sainsmart 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)
```

AFL.automation.loading.SampleCell

Classes

667777

```
SampleCell()
```

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])
```

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.

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, ...])
```

```
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.

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()
class AFL.automation.loading.SensorCallbackThread.SimpleThresholdCB(poll, period, window=5,
                                                                                 threshold=1
     __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

AFL.automation.loading.TwoSelectorBlowoutSampleCell

Classes

```
SampleCell()

Tubing(specid, length)

TwoSelectorBlowoutSampleCell(pump, ...)

Young and 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

defaultdicty
```

```
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 55

catchToWaste(sampleVolume=0.0)

AFL.automation.loading.UltimusVPressureController

Classes

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

class 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

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

```
Component(name, description)

PrepType(value[, names, module, qualname, ...])

Base class for all materials
```

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

cart(v)	Datum the square root of v
Sqrt(X,I)	Return the square root of x.

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
<pre>DeckBuilderWidget()</pre>	
<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()
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_Model
     __init__()
     \textbf{send\_deck\_config}(\textit{pi\_ip='piot2'}, \textit{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])
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

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

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.

```
BaseComponent = 1
BaseMixture = 2
Solute = 3
Solvent = 4
Solution = 5
```

AFL.automation.prepare.PrepareWidget

Functions

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

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.

class AFL.automation.prepare.PrepareWidget.PrepareWidget

AFL.automation.prepare.SampleSeriesWidget

Functions

sqrt(x,/)	Return the square root of x.
	*

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.

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)
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.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 <i>value</i> 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.

```
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.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_ternary_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])
```

 ${\bf class} \ \, {\tt AFL.automation.sample_env.TemperatureDeck}. \\ {\bf TemperatureDeck} ({\it overrides=None}) \\$

```
defaults = {'serial_port': '/dev/ttyACM0', '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()
```

6.1. API Reference 69

AFL.automation.shared

Modules

DataLabelerWidget

DatasetWidget

DiffractionLabeler

MutableQueue

PersistentConfig

ServerDiscovery

exceptions

serialization

units

utilities

widgetui

AFL.automation.shared.DataLabelerWidget

Functions

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

Classes

```
DataLabelerWodel(dataset)

DataLabelerView()

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

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

Encode categorical features as an integer array.
```

```
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)
```

6.1. API Reference 71

AFL.automation.shared.DatasetWidget

Classes

```
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, string: s
```

```
__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])
```

6.1. API Reference 73

```
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\mid 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

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

Classes

```
DiffractionLabeler(saxs_data, ...)

DiffractionLabelerModel(saxs_data, ...)

DiffractionLabelerView()

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

Encode categorical features as an integer array.
```

```
__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

6.1. API Reference 75

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 date-time_key_format

get_historical_values(key, convert_to_datetime=False)

Convenience method for gathering historical values of a parameter

AFL.automation.shared.ServerDiscovery

6.1. API Reference 77

Classes

<pre>AsyncServiceBrowser(zeroconf, type_[,])</pre>	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,])	
<pre>Zeroconf([interfaces, unicast, ip_version,])</pre>	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.

```
__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

exception AFL.automation.shared.exceptions.EmptyException

Raised when a mixture runs out of mass or volume

$\textbf{exception} \hspace{0.2cm} \textbf{AFL.automation.shared.exceptions.} \textbf{Mixing} \textbf{Exception} \\$

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

6.1. API Reference 79

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)
```

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

enforce_units(value, unit_type)	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.has_units(value)

```
AFL.automation.shared.units.is_wolume(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.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

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

6.1. API Reference 81

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

CastingServer(prep_url[, overrides])	
<pre>Client([ip, port, username, interactive]) Driver(name[, defaults, overrides])</pre>	Communicate with APIServer
<pre>OT2Client([ip, port, username, interactive])</pre>	Communicate with AFL-automation server on OT-2

class AFL.automation.sample.CastingServer.CastingServer(prep_url, overrides=None)

```
defaults = {'manifest': '/home/afl642/', 'manifest_cast': '/home/afl642/',
    '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
```

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

Create a new access token.
Create a new refresh token.
In a protected endpoint, this will return the identity of
the JWT that is accessing the endpoint.
Serialize the given arguments as JSON, and return a
Response object with the application/json mime-
type.
• 1
A decorator to protect a Flask endpoint with JSON Web
Tokens.
Render a template by name with the given context.
Send the contents of a file to the client.
Modifiy a Flask Response to set a cookie containing the
access JWT.
Modifiy a Flask Response to set a cookie containing the
refresh JWT.
Convert a string representation of truth to true (1) or false
(0).
Modifiy a Flask Response to delete the cookies contain-
ing access or refresh JWTs.

AFL.automation.APIServer.APIServer.create_access_token

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)

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() \rightarrow 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.
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

AFL.automation.APIServer.APIServer.APIServer

Methods

```
__init__(name[, data, experiment, contact, ...])
add_standard_routes()
add_unqueued_routes()
advertise_zeroconf(**kwargs)
clear_history()
clear_queue()
create_queue(driver[, add_unqueued])
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()
get_driver_object()
get_info()
                                                  Live, status page of the robot
get_queue()
get_queue_iteration()
get_queued_commands()
                                                  Return the functions, params, and defaults to be
get_quickbar()
                                                  shown in this server's quickbar
get_server_time()
get_unqueued_commands()
```

continues on next page

Table 1 – continued from previous page

	a from previous page
halt()	
index()	Live, status page of the robot
<pre>index_new()</pre>	Live, status page of the robot
init()	
<pre>init_logging([toaddrs])</pre>	
is_server_live()	
login()	
<pre>login_test()</pre>	
<pre>move_item()</pre>	
pause()	
<pre>query_driver()</pre>	
<pre>queue_state()</pre>	
remove_item()	
remove_items()	
<pre>render_unqueued(func, kwargs_add, **kwargs)</pre>	Convert an unqueued return item into web-suitable output
reorder_queue()	
<pre>reset_queue_daemon([driver])</pre>	
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)	
<pre>run_threaded([start_thread])</pre>	
<pre>send_1d_plot(result[, multi])</pre>	
send_array_as_jpg(array[, log_image,])	
<pre>set_driver_object()</pre>	
webapp()	Live, status page of the robot

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

retrieve_obj()

```
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.

1 Note

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

• vary_header (bool) – If True, the header Vary: Origin will be returned as per the W3 implementation guidelines.

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.

__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.

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.

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 __init__.py 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.
<pre>add_template_global(f[, name])</pre>	Register a custom template global function.
<pre>add_template_test(f[, name])</pre>	Register a custom template test.
<pre>add_url_rule(rule[, endpoint, view_func,])</pre>	Register a rule for routing incoming requests and building URLs.
<pre>after_request(f)</pre>	Register a function to run after each request to this object.
app_context()	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.

continues on next page

Table 2 – continued from previous page

Table 2 – continued	from previous page
<pre>create_global_jinja_loader()</pre>	Creates the loader for the Jinja2 environment.
<pre>create_jinja_environment()</pre>	Create the Jinja environment based on
	jinja_options 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"].
dispatch_request()	Does the request dispatching.
do_teardown_appcontext([exc])	Called right before the application context is popped.
do_teardown_request([exc])	Called after the request is dispatched and the response
do_teardown_request([exc])	is returned, right before the request context is popped.
and a start (and a start)	1 11
endpoint(endpoint)	Decorate a view function to register it for the given
(6	endpoint.
ensure_sync(func)	Ensure that the function is synchronous for WSGI
	workers.
<pre>errorhandler(code_or_exception)</pre>	Register a function to handle errors by code or excep-
	tion class.
<pre>finalize_request(rv[, from_error_handler])</pre>	Given the return value from a view function this final-
	izes 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 ex-
	ception catching and error handling.
<pre>get(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["GET"].
<pre>get_send_file_max_age(filename)</pre>	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_url_build_error(error, endpoint, val-	Called by <i>url_for()</i> if a BuildError was raised.
ues)	•
handle_user_exception(e)	This method is called whenever an exception occurs
* ()	that should be handled.
<pre>inject_url_defaults(endpoint, values)</pre>	Injects the URL defaults for the given endpoint di-
	rectly into the values dictionary passed.
<pre>iter_blueprints()</pre>	Iterates over all blueprints by the order they were reg-
	istered.
log_exception(exc_info)	Logs an exception.
make_aborter()	Create the object to assign to aborter.
make_config([instance_relative])	Used to create the config attribute by the Flask con-
make_confrag([motanee_relative])	structor.
<pre>make_default_options_response()</pre>	This method is called to create the default OPTIONS
make_defauft_options_fesponse()	
make waspense(m)	response.
make_response(rv)	Convert the return value from a view function to an
wales shall someone	instance of response_class.
<pre>make_shell_context()</pre>	Returns the shell context for an interactive shell for
	this application.
<pre>open_instance_resource(resource[, mode])</pre>	Opens a resource from the application's instance
	folder (instance_path).
<pre>open_resource(resource[, mode])</pre>	Open a resource file relative to <i>root_path</i> for read-
	ing.
<pre>patch(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["PATCH"].
	continues on next page

continues on next page

Table 2 – continued from previous page

Table 2 – Continued	
post(rule, **options)	Shortcut for route() with methods=["POST"].
<pre>preprocess_request()</pre>	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.
<pre>put(rule, **options)</pre>	Shortcut for <i>route()</i> with methods=["PUT"].
<pre>raise_routing_exception(request)</pre>	Intercept routing exceptions and possibly do something else.
<pre>redirect(location[, code])</pre>	Create a redirect response object.
<pre>register_blueprint(blueprint, **options)</pre>	Register a Blueprint on the application.
register_error_handler(code_or_exception, f)	Alternative error attach function to the <i>errorhandler()</i> 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.
<pre>run([host, port, debug, load_dotenv])</pre>	Runs the application on a local development server.
select_jinja_autoescape(filename)	Returns True if autoescaping should be active for the given template name.
<pre>send_static_file(filename)</pre>	The view function used to serve files from static_folder.
shell_context_processor(f)	Registers a shell context processor function.
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::.
test_cli_runner(**kwargs)	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.
<pre>trap_http_exception(e)</pre>	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.
	continues on next page

continues on next page

Table 2 – continued from previous page

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.
logger	A standard Python Logger for the app, with the same
	name as name.
name	The name of the application.
permanent_session_lifetime	A timedelta which is used to set the expiration date
	of a permanent session.
propagate_exceptions	Returns the value of the 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-
	fault FlaskCliRunner that is used by
	test_cli_runner().
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.
json	Provides access to JSON methods. continues on next page

continues on next page

Table 3 – continued from previous page

url_build_error_handlers A list of functions that are called handle_url_build_error() when url_f raises a BuildError. before_first_request_funcs A list of functions that will be called at the beging 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 she run when a shell context is created. blueprints Maps registered blueprint names to blueprint object attered. extensions a place where extensions can store application cific state. url_map The Map for this instance. You can use this to chapter the routing converters after the class was created before any routes are connected. Example::. import_name The name of the package or module that this complete that the package or module that this complete that this complete that the package or module that this	ation and jects. spe-
of the first request to this instance. **Leardown_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 she run when a shell context is created. **Blueprints** **Maps registered blueprint names to blueprint object a place where extensions can store application cific state. **url_map** **The Map for this instance. You can use this to che the routing converters after the class was created before any routes are connected. Example::. **import_name** **The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created before any routes are connected. The name of the package or module that this context is created.	ation nould jects.
context is destroyed. Shell_context_processors A list of shell context processor functions that sl be run when a shell context is created. Maps registered blueprint names to blueprint ob a place where extensions can store application cific state. Url_map The Map for this instance. You can use this to che routing converters after the class was created before any routes are connected. Example::. import_name The name of the package or module that this context.	jects. spe-
be run when a shell context is created. blueprints Maps registered blueprint names to blueprint observers application cific state. url_map The Map for this instance. You can use this to che the routing converters after the class was created before any routes are connected. Example::. import_name The name of the package or module that this content is created.	jects. spe-
a place where extensions can store application cific state. url_map The Map for this instance. You can use this to che the routing converters after the class was created before any routes are connected. Example::. import_name The name of the package or module that this content is the content of the package or module that this content is the content of the package or module.	spe-
a place where extensions can store application cific state. url_map The Map for this instance. You can use this to che the routing converters after the class was created before any routes are connected. Example::. import_name The name of the package or module that this content is the content of the package or module that this content is the content of the package or module.	spe-
the routing converters after the class was create before any routes are connected. Example::. import_name The name of the package or module that this content is a second of the package or module.	
	·
template_folder The path to the templates folder, relative root_path, to add to the template loader.	e to
root_pathAbsolute path to the package on the filesystem.cliThe Click command group for registering CLI mands for this object.	
view_functions A dictionary mapping endpoint names to view tions.	func-
error_handler_spec A data structure of registered error hand in the format {scope: {code: {cl handler}}}.	ilers, ass:
before_request_funcs A data structure of functions to call at the ginning of each request, in the format {sc [functions]}.	
after_request_funcsA data structure of functions to call at the end ofrequest, in the format {scope: [functions]	
teardown_request_funcs A data structure of functions to call at the end of request even if an exception is raised, in the forest even [functions].	rmat
template_context_processors A data structure of functions to call to pass extra text values when rendering templates, in the form {scope: [functions]}.	rmat
url_value_preprocessorsA data structure of functions to call to modify the word arguments passed to the view function, i format {scope: [functions]}.	-
url_default_functionsA data structure of functions to call to modify the word arguments when generating URLs, in the for {scope: [functions]}.	

request_class

alias of Request

${\tt response_class}$

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 *json_provider_class* 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</pre> object> the session interface to use. By default an instance of SecureCookieSessionInterface is used here.

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)

__init__(import name: str, static url path: str | None = None, static folder: str | PathLike | None = 'static',

Added in version 0.8.

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() → 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) \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.

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):
```

(continues on next page)

110

(continued from previous page)

```
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() → 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 endpoint() 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) \rightarrow 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.

```
shell\_context\_processor(f: T\_shell\_context\_processor) \rightarrow 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[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]*) → None Logs an exception. This is called by *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.

finalize_request(rv: 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 | HTTPException, from_error_handler: bool =

False) → Response

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

$\textbf{make_default_options_response()} \rightarrow Response$

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) \rightarrow 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]) \rightarrow 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 \mid None = None, _method: str \mid None = None, _scheme: str \mid None = None, _external: str \mid None = None, **values: str \mid None = None, _scheme: str \mid None = None, _s
```

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 <code>handle_url_build_error()</code> method is called. If that returns a string, that is returned, otherwise a <code>BuildError</code> 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 | 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) → 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 isonify'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().

```
\begin{tabular}{ll} \textbf{delete}(\textit{rule: str}, **options: Any) \rightarrow Callable[[T\_route], T\_route] \\ Shortcut for \textit{route()} with methods=["DELETE"]. \\ \end{tabular}
```

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")
```

(continues on next page)

(continued from previous page)

```
def example():
    ...
```

Parameters

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

```
\label{eq:content} \begin{split} \textbf{get}(\textit{rule: str}, **options: Any}) \rightarrow & \text{Callable}[[T\_route], T\_route] \\ & \text{Shortcut for } \textit{route()} \text{ with methods=["GET"]}. \end{split}
```

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".

Shortcut for *route()* with methods=["POST"].

Added in version 2.0.

$$\label{eq:content} \begin{split} \text{put}(\textit{rule: str}, **options: Any}) \rightarrow \text{Callable}[[T_route], T_route] \\ \text{Shortcut for } \textit{route()} \text{ with methods=["PUT"]}. \end{split}$$

Added in version 2.0.

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[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

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 /reqcontext.

Typically you should not call this from your own code. A request context is automatically pushed by the $wsgi_app()$ when handling a request. Use $test_request_context()$ to create an environment and context instead of this method.

Parameters

environ - a WSGI environment

```
test_request_context(*args: Any, **kwargs: Any) → 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
```

```
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.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 | 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.

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.
<pre>additional_headers_loader(callback)</pre>	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 UN-
	VERIFIED 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 encoun-
manda Carab talam I andam(anliberta)	tered.
needs_fresh_token_loader(callback)	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
- 0. 0.101 _ 00.101 00.101 _ (0.1110 . 1011)	a custom response when a revoked token is encountered.
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 con-
	vert a JWT into a python object that can be used in a protected endpoint.
	i i

__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) → 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.

$user_identity_loader(callback: Callable) \rightarrow Callable$

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

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

Bases: object

```
__init__(*filters)
```

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.

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

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(mailhost, fromaddr, toaddrs, subject)	Initialize the handler.
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.
format(record)	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.
set_name(name)	

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

init(*args, **kwargs)	
addresses_by_version(version)	List addresses matching IP version.
async_clear_cache()	Clear the cache for this service info.
<pre>async_request(zc, timeout[, question_type,])</pre>	Returns true if the service could be discovered on the 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 discovered.
<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.
name
                                                 The name of the service.
                                                 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 | 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.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_listener(listener, question)	Adds a listener for a given question.

continues on next page

Table 4 – continued from previous page

add service_listener(typelistener) Adds a listener for a particular service type. async_check_service(info, allow_name_change) Adds a listener for a given question. async_get_service_info(type_, name ,) Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique. async_notify_all() Returns network service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds. async_remove_listener(listener) Schedule an async_notify_all. async_wair_tore(info) Removes a listener. async_wair_tore(info) Secnds an outgoing packet. unregister aservice(info) Unregister aservice. async_wait_for_start([timeout]) Calling task waits for a given number of milliseconds or until notified. async_wait_for_start([timeout]) 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. generate_service_upery(info) Generate a pury to lookup a service. get_service_info(type_, name[, timeout,]) Resisters servicing further queries. get_service_info(type_, name[, timeout,]) Notifies all waiting threads and notify listeners. get_service_info(type_, name[, timeout,]) Notifies all waiting threads and notify lis	Table 4 – Continued	
async_get_service(info, allow_name_change) async_get_service_info(type_, name[,]) async_get_service_info(type_, name[,]) async_notify_all() async_register_service(info[, ttl,]) async_register_service(info[, ttl,]) async_send(out[, addr, port, v6_flow_scope,]) async_urregister_all_services() async_urregister_service(info) async_urregister_service(info) async_urregister_service(info) async_urregister_service(info) async_urregister_service(info) async_urregister_service(info) async_update_service(info) async_wait_for_start([timeout]) async_wait_for_start([timeout]) async_wait_for_start([timeout]) close() generate_service_depery(info) generate_service_depery(info) generate_service_depery(info) generate_urregister_all_services() get_service_info(type_, name[, timeout,]) async_wait_for_start([timeout]) close() generate_service_depery(info) generate_service_depery(info) generate_service_depery(info) generate_urregister_all_services() get_service_info(type_, name[, timeout,]) async_wait_for_start([timeout]) close() finds the background threads, and prevent this instance from servicing further queries. Generate a purey to lookup a service. Generate a purey to lookup 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_warning(*logger_data) log_exception_	<pre>add_service_listener(type_, listener)</pre>	Adds a listener for a particular service type.
ifying the ServiceInfo passed in if it is not unique. 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. Schedule an async_notify_all. Registers service information to the network with a default TTL. Removes a listener. Sends an outgoing packet. Unregister a service. Sends an outgoing packet. Unregister a service. Unregister all registered services. Unregister a service. Registers service information to the network with a default TTL. Calling task waits for a given number of milliseconds or until notified. Async_wait_for_start([timeout]) Async_wait_for_start([timeout]) Sends the background threads, and prevent this instance from servicing further queries. Generate a proadcast 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. In the part of the part o	<pre>async_add_listener(listener, question)</pre>	Adds a listener for a given question.
async_notify_all() async_register_service(info[, ttl,]) async_remove_listener(listener) async_unregister_service(info] async_unregister_service(info) async_wait(timeout) close() generate_service_uref(info) generate_service_uref(info) generate_service_uref(info) generate_info(type_, name[, timeout,]) get_service_info(type_, name[, timeout,]) async_cmove_listener(listener) async_wait(limeout) close() generate_service_duref(info) generate_service_fundebug(*logger_data) log_exception_once(exc, *args) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_service_listener(istener) remove_all_service_listener(istener) remove_listener(listener) send(out[, addr, port, v6_flow_scope, transport[)] start() unregister_service(info) unregister_service(info) Unregister a service. Registers service information to the network with a default TTL. Removes a listener. Register information to the network with a default TTL. Removes a listener from the set that is currently listening. Register 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 a service. Unregister a service. Unregister a service. Unregister a service. Service information to the network with a default TTL. Removes a listener. Registers service information to the network with a default TTL. Removes a listener. Registers service information to the network with a default TTL. Removes a listener. Register servic	<pre>async_check_service(info, allow_name_change)</pre>	Checks the network for a unique service name, mod-
async_notify_all() async_register_service(info[, ttl,]) async_remove_listener(listener) async_send(outl, addr, port, v6_flow_scope,]) async_unregister_all_services() async_unregister_service(info) async_wait(timeout) async_wait_for_start([timeout]) log_exception_debug(*logger_data) log_exception_debug(*logger_data) log_exception_debug(*logger_data) log_exception_lore(*args) name and type, or None if no service matches by the timeout, which defaults to 3 seconds. Schedule an async_notify_all. Registers service information to the network with a default TTL. 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 a broadcast to announce a service. Generate a puery to lookup a service. Generate a puery to lookup a service. Generate a query to lookup a service. Generate a query to lookup 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. Removes a listener from the set that is currently listening. Sends an outgoing packet. Sends an outgoing packet threadsafe. Start Zeroconf. Unregister a service.		ifying the ServiceInfo passed in if it is not unique.
async_notify_all() async_register_service(info[, ttl,]) async_remove_listener(listener) async_send(outl, addr, port, v6_flow_scope,]) async_unregister_lservice(info) async_unregister_service(info) async_unregister_service(info) async_unregister_service(info) async_wait(timeout) async_wait[for_start([timeout]) close() async_wait_ere_service_dunfo) generate_service_query(info) generate_unregister_all_services() generate_unregister_service(info], ttl,]) generate_nnregister_all_services() generate_unregister_all_services() generate_unregister_all_services() unregister_all_services()	<pre>async_get_service_info(type_, name[,])</pre>	Returns network's service information for a particular
timeout, which defaults to 3 seconds. Schedule an async_notify_all() async_remove_listener(listener) async_send(out[, addr, port, v6_flow_scope,]) async_unregister_all_services() async_unregister_service(info) async_wait(timeout) async_wait(timeout) async_wait_for_start([timeout]) generate_service_dure() generate_service_query(info) generate_service_query(info) generate_service_query(info) generate_service_info(type_, name[, timeout,]) generate_service_info(type_, name[, timeout,]) get_service_tin_debug(*logger_data) log_exception_debug(*logger_data) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_service_listener) remove_all_service_listenery remove_service_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_services() unreg		name and type, or None if no service matches by the
async_register_service(info[, ttl,]) async_remove_listener(listener) async_send(out[, addr, port, v6_flow_scope,]) async_unregister_all_services() async_unregister_service(info) async_unregister_service(info) async_unregister_service(info) async_wait(timeout) close() async_wait_for_start([timeout]) generate_service_broadcast(info, ttl[,]) generate_service_query(info) generate_unregister_all_services() agenerate_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_service_listener) remove_listener(listener) remove_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_services() unregister_service. infontion to the network with a default TTL. Removes a listener. Removes a listener. Removes a listener. Removes a listener information to the network with a default TTL. Removes a listener information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.		
async_register_service(info[, ttl,]) async_remove_listener(listener) async_send(out[, addr, port, v6_flow_scope,]) async_unregister_all_services() async_unregister_service(info) async_wait(timeout) async_wait(timeout) async_wait_for_start([timeout]) generate_service_unery(info) generate_service_info(type_, name[, timeout,]) log_exception_once(exc, *args) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_start(listener) remove_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_service. Registers service information to the network with a default TTL. Removes a listener. Sends an outgoing packet. Unregister all registered services. Unregister alle	async_notify_all()	
default TTL. Removes a listener. async_remove_listener(listener) async_unregister_all_services() async_unregister_service(info) async_update_service(info) async_wait(timeout) async_wait_for_start([timeout]) close() 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() remove_listener(listener) remove_listener(listener) remove_service_listener(listener) remove_service_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_service. log-exception_seder(info) default TTL. Removes a listener. Sends an outgoing packet. Unregister all registered services. Unregister a		
async_send(out[, addr, port, v6_flow_scope,]) Sends an outgoing packet. async_unregister_all_services() Unregister all registered services. async_update_service(info) Registers service information to the network with a default TTL. async_wait(timeout) Calling task waits for a given number of milliseconds or until notified. async_wait_for_start([timeout]) 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. generate_service_unery(info) Generate a broadcast to announce a service. generate_service_query(info) Generate a pury to lookup a service. 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) Notifies all waiting threads and notify listeners. 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.		default TTL.
async_unregister_all_services() Unregister all registered services. async_unpdate_service(info) Unregister a service information to the network with a default TTL. async_wait(timeout) Calling task waits for a given number of milliseconds or until notified. async_wait_for_start([timeout]) 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. generate_service_broadcast(info, ttl[,]) Generate a broadcast to announce a service. generate_service_query(info) Generate a query to lookup a service. generate and incompanies. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast to announce a service. Generate a proadcast. Generate a proadcast to announce a service. Generate a proadcast. Generate a proadcast to announce a service. <		
async_unregister_service(info) async_wait(timeout) async_wait(timeout) async_wait_for_start([timeout]) async_wait_for_start([timeout]) close() generate_service_broadcast(info, ttl[,]) generate_service_unery(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_exception_warning(*logger_data) log_exception_service_listener() remove_all_service_listener() remove_service_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) saync_wait for		
async_wait(timeout) async_wait(timeout) async_wait(timeout) close() close() close() generate_service_broadcast(info, ttl[,]) generate_service_duery(info) generate_unregister_all_services() default TTL. 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 a broadcast 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_once(exc, *args) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_warning_once(*args) notify_all() remove_all_service_listeners() Removes a listener from the set that is currently listening. remove_service_listener(listener) remove_service_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_service(info) Unregister all registered service. Unregister as ervice information to the network with a default TTL. Removes a listener. Removes a listener from the set that is currently listening. Sends an outgoing packet threadsafe. Start Zeroconf. Unregister_all_services() Unregister all registered services. Unregister avervice.		
default TTL. async_wait(timeout) async_wait_(for_start([timeout]) async_wait_for_start([timeout]) 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,]) log_exception_debug(*logger_data) log_exception_once(exc, *args) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_service_listeners() remove_listener(listener) remove_service_listener(listener) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_service(info) Unregister as vice. Unregister a given number of milliseconds or until notified. Wait for start up for actions that require a running Ze-roconf instance. Ends the background threads, and prevent this instance from servicing further queries. Generate a DNSOutpoing 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. Sends an outgoing packet threadsafe. Start Zeroconf. unregister_all_services() Unregister all registered services. unregister_service(info) Unregister as vervice.		
or until notified. async_wait_for_start([timeout]) Close() Ends the background threads, and prevent this instance from servicing further queries. generate_service_duery(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() remove_all_service_listener(service_listener(service	async_update_service(info)	
roconf instance. 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,]) 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_warning_once(*args) notify_all() register_service(info[, 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) Unregister a service. Unregister a service.	<pre>async_wait(timeout)</pre>	
Ends the background threads, and prevent this instance from servicing further queries. generate_service_proadcast(info, ttl[,]) Generate a broadcast to announce a service. 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_warning(*logger_data) log_exception_warning(*logger_data) log_exception_warning(*logger_data) log_exception_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 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 registered services. unregister_service(info) Unregister all registered services. unregister_service(info) Unregister a service.	<pre>async_wait_for_start([timeout])</pre>	•
stance from servicing further queries. generate_service_broadcast(info, ttl[,]) generate_service_query(info) generate a query to lookup a service. generate a DNSOutgoing goodbye for all services and remove them from the registry. 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_exception_warning(*logger_data) log_exception_service(info[, ttl,]) 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]) send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_services() Unregister all_registered services. Unregister a service.	close()	
generate_service_broadcast(info, ttl[,]) Generate a broadcast to announce a service. generate_service_query(info) Generate a Query to lookup a service. 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_warning(*logger_data) 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() Unregister all registered services. unregister_service(info) Unregister as service.	V	
generate_service_query(info) generate_unregister_all_services() 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_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]) start() unregister_all_services() unregister_all_services() unregister_all registered services. unregister_service(info) Unregister a service.	generate service broadcast(info.ttl[])	
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 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_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_service(info) Unregister a service.		* *
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) log_exception_warning(*logger_data) log_warning_once(*args) 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_aservice.	generate_unregister_urr_services()	
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()	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() 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.	β,, ,,, ,	name and type, or None if no service matches by the
log_exception_warning(*logger_data) log_warning_once(*args) notify_all()	<pre>log_exception_debug(*logger_data)</pre>	
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.	<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.	<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.	log_warning_once(*args)	
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.	notify_all()	Notifies all waiting threads and notify listeners.
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.		Registers service information to the network with a
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.	remove_all_service_listeners()	Removes a listener from the set that is currently lis-
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.	remove listener(listener)	
tening. send(out[, addr, port, v6_flow_scope, transport]) start() unregister_all_services() unregister_service(info) tening. Sends an outgoing packet threadsafe. Start Zeroconf. Unregister all registered services. Unregister a service.		
start()Start Zeroconf.unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.		tening.
unregister_all_services()Unregister all registered services.unregister_service(info)Unregister a service.		
unregister_service(info) Unregister a service.	· ·	
update_service(info) Registers service information to the network with a		
default TTL.	<pre>update_service(info)</pre>	

Attributes

listeners

started

Check if the instance has started.

```
__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)

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 DNSQuestion-Type.QU)

 $\begin{tabular}{ll} \begin{tabular}{ll} \beg$

Generate a broadcast to announce a service.

generate_service_query(*info:* ServiceInfo) → 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.

 $generate_unregister_all_services() \rightarrow DNSOutgoing | None$

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

 $\textbf{async_add_listener}(\textit{listener}: \textit{RecordUpdateListener}, \textit{question}: \textit{DNSQuestion} \mid \textit{list[DNSQuestion]} \mid \textit{None}) \\ \rightarrow \textit{None}$

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 | None = None, port: int = 5353, v6_flow_scope: tuple[()] | tuple[int, int]
            = (), transport: \_WrappedTransport \mid 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.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 uuid 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()

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])

enqueued_base(**kwargs)

from_server_name(server_name, **kwargs)

get_config(name[, print_console, interactive])

get_driver_object(name)

continues on next page
```

Table 5 – continued from previous page

```
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_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.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>	
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.
sa_discover_server_by_name(service_name)	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)	
<pre>makeRegistrar()</pre>	
sqrt(x, l)	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()
 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
               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.Driver.PersistentConfig
class AFL.automation.APIServer.Driver.PersistentConfig(path, defaults=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.

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
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.
- $\hbox{\bf \bullet datetime_key} \ (str, \ optional) datetime \ formatted \ string \ as \ defined \ by \ datetime_key_format \\$

get_historical_values(key, convert_to_datetime=False)

Convenience method for gathering historical values of a parameter

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 $\geq x$.

AFL.automation.APIServer.DummyDriver.listify

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

AFL.automation.APIServer.DummyDriver.sqrt

```
AFL.automation.APIServer.DummyDriver.\mathbf{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 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()
                                                   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)

Methods

```
__init__([name, overrides])
deposit_obj(obj[, uid])
                                                   Store an object in the dropbox
dummy_reset_tank_levels([rinsel,
                                          rinse2,
waste])
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()
how_many(**kwargs)
loadSample([cellname, sampleVolume])
post_execute(**kwargs)
                                                   Executed after each call to execute
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])
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,/)	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])
```

AFL.automation.APIServer.DummyOT2Driver.Driver

```
class AFL.automation.APIServer.DummyOT2Driver.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
               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.DummyOT2Driver.DummyDriver
class AFL.automation.APIServer.DummyOT2Driver.DummyDriver(name=None, overrides=None)
     Bases: Driver
     __init__(name=None, overrides=None)
```

Methods

init([name, overrides])	
<pre>deposit_obj(obj[, uid]) dummy_reset_tank_levels([rinse1, rinse waste])</pre>	Store an object in the dropbox 2,
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_prep_target(**kwargs)</pre>	
<pre>get_sample()</pre>	
how_many(**kwargs)	
<pre>loadSample([cellname, sampleVolume])</pre>	
<pre>post_execute(**kwargs) pre_execute(**kwargs) queued()</pre>	Executed after each call to execute Executed before each call to execute
quickbar()	
<pre>quickbar_test([text_field, int_field,])</pre>	
quickbar_test2()	
reset_sample()	
<pre>retrieve_obj(uid[, delete]) set_config(**kwargs)</pre>	Retrieve an object from the dropbox
set_data(data)	Set data in the DataPacket object
<pre>set_object([serialized])</pre>	
<pre>set_sample(sample_name[, sample_uuid])</pre>	
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, kwarg3])	2, A test command with positional and keyword parameters continues on next page

Table 6 – continued from previous page

```
test_image(**kwargs)

test_plot(**kwargs)

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
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
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)
     Methods
       __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.
<pre>QueueDaemon(app, driver, task_queue, history)</pre>	

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()	
<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.
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
                                                                           debug=False, data=None)
```

class AFL.automation.APIServer.QueueDaemon.**QueueDaemon**(app, driver, task_queue, history,

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

class AFL.automation.instrument.DummySAS.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()
 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
               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
```

init([overrides])	connect to spec
deposit_obj(obj[, uid])	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

```
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)
```

AFL.automation.instrument.FileCamera

Classes

```
FileCamera()
```

AFL.automation.instrument.FileCamera.FileCamera

```
class AFL.automation.instrument.FileCamera.FileCamera
Bases: object
__init__()
```

Methods

```
__init__()
collect(fname)

__init__()
collect(fname)

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

AFL.automation.instrument.l22SAXS

Classes

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

I22SAXS([overrides])
```

AFL.automation.instrument.l22SAXS.Driver

```
class AFL.automation.instrument.I22SAXS.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 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()
                                                   Executed after each call to execute
post_execute(**kwargs)
                                                   Executed before each call to execute
pre_execute(**kwargs)
queued()
quickbar()
reset_sample()
                                                   Retrieve an object from the dropbox
retrieve_obj(uid[, delete])
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])	
deposit_obj(obj[, uid])	Store an object in the dropbox
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>	
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute
queued()	
quickbar()	
<pre>read_integrated(scanid)</pre>	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
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()	

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)
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.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)
```

AFL.automation.instrument.SeabreezeUVVis

Classes

__init__(*url*)

collect()

camera_reset()

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)
```

```
__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
               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.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__()
```

init()	
absolute()	Return an absolute version of this path by prepending
. 0	the current working directory.
as_posix()	Return the string representation of the path with forward (/) slashes.
as_uri()	Return the path as a 'file' URI.
<pre>chmod(mode, *[, follow_symlinks])</pre>	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 con-
	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.
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, it
	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()	Whether this path is a socket.
is_symlink()	Whether this path is a symbolic link.
iterdir()	Iterate over the files in this directory.
joinpath(*args)	Combine this path with one or several arguments, and
	return a new path representing either a subpath (if al
	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
1:-1- +-(44)	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
	symlink's status information is returned, rather than
<pre>match(path_pattern)</pre>	its target's. Return True if this path matches the given pattern.
mkdir([mode, parents, exist_ok])	Create a new directory at this given path.
miserr ([mode, parents, exist_ok])	continues on next page

continues on next page

Table 7 – continued from previous page

open([mode, buffering, encoding, errors,]) Open the file pointed by this path and return a file 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. read_tink() 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) Make the path absolute, resolving all symlinks on the way and also normalizing it. 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, including directories) matching the given relative pattern, anywhere in this subtree. rmdir() Remove this directory. samefile(other_path) Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([miss		- 1 1 3 -
owner()Return the login name of the file owner.read_bytes()Open the file in bytes mode, read it, and close the file.read_text([encoding, errors])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, 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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.with_name(name)Return a new path with the file name changed.with_stem(stem)Return a new path with the file suffix changed.with_stem(stem)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>open([mode, buffering, encoding, errors,])</pre>	• • • • • • • • • • • • • • • • • • • •
read_bytes()Open the file in bytes mode, read it, and close the file.read_text([encoding, errors])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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.with_name(name)Return a new path with the file name changed.with_stem(stem)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.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the		1
read_text([encoding, errors])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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.with_name(name)Return a new path with the file name changed.with_stem(stem)Return a new path with the stem changed.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the file.	owner()	Return the login name of the file owner.
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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.with_name(name)Return a new path with the file name changed.with_stem(stem)Return a new path with the stem changed.with_stefix(suffix)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.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	read_bytes()	Open the file in bytes mode, read it, and close the file.
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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.with_name(name)Return a new path with the file name changed.with_stem(stem)Return a new path with the file suffix changed.with_suffix(suffix)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.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	<pre>read_text([encoding, errors])</pre>	Open the file in text mode, read it, and close the file.
the passed arguments. rename(target) Rename this path to the target path. 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. Recursively yield all existing files (of any kind, including directories) matching the given relative pattern, anywhere in this subtree. rmdir() Remove this directory. samefile(other_path) Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. with_name(name) Return a new path with the file name changed. with_stem(stem) with_suffix(suffix) 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. write_text(data[, encoding, errors, newline])	readlink()	Return the path to which the symbolic link points.
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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	relative_to(*other)	Return the relative path to another path identified by
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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the		the passed arguments.
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, including directories) matching the given relative pattern, anywhere in this subtree. rmdir() Remove this directory. samefile(other_path) Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. with_name(name) Return a new path with the file name changed. with_stem(stem) Return a new path with the file suffix changed. with_suffix(suffix) Return a new path with the file suffix changed. open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the	rename(target)	Rename this path to the target path.
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, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	replace(target)	Rename this path to the target path, overwriting if that
way and also normalizing it. rglob(pattern) Recursively yield all existing files (of any kind, including directories) matching the given relative pattern, anywhere in this subtree. rmdir() Remove this directory. samefile(other_path) Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. with_name(name) with_stem(stem) Return a new path with the file name changed. with_suffix(suffix) Return a new path with the stem changed. with_suffix(suffix) Return a new path with the file suffix changed. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the		path exists.
rglob(pattern)Recursively yield all existing files (of any kind, including directories) matching the given relative pattern, anywhere in this subtree.rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	resolve([strict])	Make the path absolute, resolving all symlinks on the
cluding directories) matching the given relative pattern, anywhere in this subtree. rmdir() Remove this directory. Return whether other_path is the same or not as this file (as returned by os.path.samefile()). Stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. Symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. 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. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the		way and also normalizing it.
tern, anywhere in this subtree. rmdir() samefile(other_path) Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. with_name(name) Return a new path with the file name changed. with_stem(stem) with_stem(stem) Return a new path with the stem changed. with_suffix(suffix) Return a new path with the file suffix changed. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the	rglob(pattern)	Recursively yield all existing files (of any kind, in-
rmdir()Remove this directory.samefile(other_path)Return whether other_path is the same or not as this file (as returned by os.path.samefile()).stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the		cluding directories) matching the given relative pat-
Return whether other_path is the same or not as this file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. 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. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the		tern, anywhere in this subtree.
file (as returned by os.path.samefile()). stat(*[, follow_symlinks]) Return the result of the stat() system call on this path, like os.stat() does. symlink_to(target[, target_is_directory]) 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. unlink([missing_ok]) Remove this file or link. 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. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the	rmdir()	Remove this directory.
stat(*[, follow_symlinks])Return the result of the stat() system call on this path, like os.stat() does.symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	<pre>samefile(other_path)</pre>	Return whether other_path is the same or not as this
like os.stat() does. symlink_to(target[, target_is_directory]) touch([mode, exist_ok]) Create this file with the given access mode, if it doesn't exist. unlink([missing_ok]) Remove this file or link. with_name(name) Return a new path with the file name changed. with_stem(stem) Return a new path with the file suffix changed. with_suffix(suffix) Return a new path with the file suffix changed. Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the		
symlink_to(target[, target_is_directory])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.unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	<pre>stat(*[, follow_symlinks])</pre>	Return the result of the stat() system call on this path,
touch([mode, exist_ok]) Create this file with the given access mode, if it doesn't exist. unlink([missing_ok]) Remove this file or link. 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. write_bytes(data) Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the		like os.stat() does.
doesn't exist. unlink([missing_ok]) Remove this file or link. 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. write_bytes(data) Open the file in bytes mode, write to it, and close the file. write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the	<pre>symlink_to(target[, target_is_directory])</pre>	Make this path a symlink pointing to the target path.
unlink([missing_ok])Remove this file or link.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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	touch([mode, exist_ok])	Create this file with the given access mode, if it
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.write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the		doesn't exist.
with_stem(stem)Return a new path with the stem changed.with_suffix(suffix)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.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	<pre>unlink([missing_ok])</pre>	Remove this file or link.
with_suffix(suffix)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.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	with_name(name)	Return a new path with the file name changed.
write_bytes(data)Open the file in bytes mode, write to it, and close the file.write_text(data[, encoding, errors, newline])Open the file in text mode, write to it, and close the	<pre>with_stem(stem)</pre>	Return a new path with the stem changed.
file. write_text(data[, encoding, errors, newline]) file. Open the file in text mode, write to it, and close the	<pre>with_suffix(suffix)</pre>	Return a new path with the file suffix changed.
write_text(data[, encoding, errors, newline]) Open the file in text mode, write to it, and close the	<pre>write_bytes(data)</pre>	Open the file in bytes mode, write to it, and close the
		file.
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

```
\begin{tabular}{ll} \textbf{class} & \textbf{AFL}. \textbf{automation.instrument.SeabreezeUVVis.SeabreezeUVVis} (backend='cseabreeze', \\ & device\_serial=None, \\ & overrides=None) \end{tabular}
```

```
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])
```

6.2. Modules 203

continues on next page

Table 8 – continued from previous page

Table 8 – continued	from previous page
<pre>collectSingleSpectrum([set_reference, set_air])</pre>	
<pre>deposit_obj(obj[, uid])</pre>	Store an object in the dropbox
execute(**kwargs)	2.000 m. 00,000 m. 000 p. 000
<pre>gather_defaults()</pre>	Gather all inherited static class-level dictionaries called default.
<pre>getExposure()</pre>	
<pre>getExposureDelay()</pre>	
<pre>getFilename()</pre>	
<pre>getFilepath()</pre>	
<pre>getSaveSingleScan()</pre>	
<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_execute(**kwargs)	Executed before each call to execute
queued()	
quickbar()	
reduced(data_raw_mean, data_raw_std[,])	
reset_sample()	
<pre>retrieve_obj(uid[, delete])</pre>	Retrieve an object from the dropbox
setExposure(time)	ı
setExposureDelay(time)	
setFilename(filename)	
setFilepath(filepath)	
setSaveSingleScan(saveSingleScan)	
<pre>set_config(**kwargs)</pre>	
set_data(data)	Set data in the DataPacket object
set_object([serialized])	
	continues on next page

continues on next page

Table 8 – continued from previous page

```
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

AFL.automation.instrument.SpecScreen_Driver.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)
```

```
__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
                                                    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()
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)
```

```
__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	vided by Chemyx.
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
LoadStopperDriver	
MultiChannelRelay	
NE1kSyringePump	
OneSelectorBlowoutSampleCell	
PneumaticPressureSampleCell	
PneumaticSampleCell	
PressureController	
PressureControllerAsPump	
PushPullSelectorSampleCell	
RSoXSSolutionSampleCell	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
SyringePump	
Tubing	
TwoSelectorBlowoutSampleCell	
UltimusVPressureController	
ViciMultiposSelector	
<u>214</u>	Chapter 6. Reference

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

- 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(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

- 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)

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
                  • to
                                  (port - string describing the serial port the actuator is
                    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.
```

AFL.automation.loading.ChemyxSyringePump

query the current selected position

getPort(as_str=False)

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>	
<pre>parsePortName(portinfo)</pre>	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)

```
__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
class AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump(port, syringe_id_mm,
                                                                        syringe_volume, baud=9600,
                                                                        flow delay=5)
     Bases: SyringePump
     __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
```

```
__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()
wait_dosage_finished([timeout_seconds])
                                                    The function waits until the last dosage command has
                                                    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()
```

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.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()
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)
             On macOS and Linux, selects only usbserial options and parses the 8 character serial number.
\textbf{class} \  \, \textbf{AFL}. automation.loading. ChemyxSyringePump. \textbf{ChemyxConnection} (port, baudrate, x=0, mode=0, mode=
                                                                                                                                                                                             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

```
DigitalOutPressureController(digital_out, ...)

PressureController()

Abstract superclass for pressure controllers that provides timed dispensing
```

AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController

 $\textbf{class AFL}. automation. loading. Digital OutPressureController. \textbf{DigitalOutPressureController} (digital_out, pressure_to_v_conv)$

init(digital_out, pressure_to_v_conv)		Initializes a DigitalOutPressureController
<pre>blockUntilStatusStopped([pollingdelay])</pre>		block execution until the controller finishes a dis-
		pense
dispenseRunning()		Returns true if a timed dispense is running, false oth-
		erwise.
<pre>ramp_dispense(dispense_start_pressure,)</pre>		Perform a pressure dispense with a linear ramp in
		pressure between dispense_start_pressure and dis-
		pense_stop_pressure, stopping after dispense_time.
<pre>set_P(pressure)</pre>		
stop()		Abort the current timed dispense action.
timed_dispense(dispense_pressure,	dis-	Perform a pressure dispense at pressure dis-
pense_time)		pense_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

class AFL.automation.loading.DigitalOutPressureController.PressureController

Bases: object

Abstract superclass for pressure controllers that provides timed dispensing

__init__()

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.

 $\textbf{class AFL}. \textbf{automation.} loading. \textbf{DigitalOutPressureController.} \textbf{DigitalOutPressureController}. \textbf{DigitalOutPressureContro$

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, ...])
```

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

- 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}

```
AFL-automation
     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
```

class AFL.automation.loading.DoubleViciMultiposSelector.ViciMultiposSelector(port, baudrate=9600, portlabels=None)

```
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}

```
__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)

class AFL.automation.loading.DoubleViciMultiposSelector.DoubleViciMultiposSelector(port1,

port2, baudrate=9600, portla-

bels=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.
<pre>blockUntilStatusStopped([pollingdelay])</pre>	
dispense(volume[, block, delay])	
<pre>emptySyringe()</pre>	
<pre>getRate()</pre>	
<pre>getStatus()</pre>	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.
<pre>withdraw(volume[, block, delay])</pre>	

```
__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
class AFL.automation.loading.DummyPump.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.DummyPump.SyringePump
class AFL.automation.loading.DummyPump.SyringePump
     Bases: object
     __init__()
```

```
__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()
```

${\bf AFL}. automation. Io ading. Flow Selector. Flow Selector$

```
class AFL.automation.loading.FlowSelector.FlowSelector
    Bases: object
    __init__()
```

Methods

```
__init__()

getPort()

selectPort()
```

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

AFL.automation.loading.LoadStopperDriver

Classes

selectPort()

Client([ip, port, username, interactive])	Communicate with APIServer
Driver(name[, defaults, overrides])	
<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])
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_commmands([inherit_commands])
halt()
```

continues on next page

Table 9 – continued from previous page

```
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.loading.LoadStopperDriver.Driver
class AFL.automation.loading.LoadStopperDriver.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()
 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)
```

overrides=None, data=None,

name='LoadStopperDriver')

sensorlabel=",

```
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.LoadStopperDriver.LoadStopperDriver
class AFL.automation.loading.LoadStopperDriver.LoadStopperDriver(sensor, load_client=None,
                                                                               load object=None,
                                                                               auto_initialize=True,
```

Bases: Driver

Driver for stopping loads

__init__(sensor, load_client=None, load_object=None, auto_initialize=True, overrides=None, data=None, sensorlabel=", name='LoadStopperDriver')

```
__init__(sensor[, load_client, load_object, ...])
calibrate_sensor()
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()
post_execute(**kwargs)
                                                   Executed after each call to execute
                                                   Executed before each call to execute
pre_execute(**kwargs)
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)
```

Driver for stopping loads

calibrate_sensor()

```
read_sensor()
    read_poll()
    read_poll_load()
    reset()
    reset_poll()
    reset_stopper()
AFL.automation.loading.MultiChannelRelay
Classes
 MultiChannelRelay()
AFL.automation.loading.MultiChannelRelay.MultiChannelRelay
class AFL.automation.loading.MultiChannelRelay.MultiChannelRelay
     Bases: object
     __init__()
     Methods
      __init__()
      getChannels(channels)
      setChannels(channels)
      toggleChannels(channels)
```

```
setChannels(channels)

getChannels(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

```
Bases: SyringePump
__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.

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>	
<pre>getStatus()</pre>	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)
     \textbf{sendCommand}(\textit{cmd}, \textit{response} = \textit{True}, \textit{questionmarkOK} = \textit{False}, \textit{timeout} = -1, \textit{debug} = \textit{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()
class AFL.automation.loading.NE1kSyringePump.NE1kSyringePump(port, syringe_id_mm,
                                                                          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

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
               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.OneSelectorBlowoutSampleCell.OneSelectorBlowoutSampleCell
```

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)
```

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

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.
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>	
	continues on next page

Table 10 – continued from previous page

```
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)
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()
setRinseLevel(vol)
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)
status()
swish(vol)
transfer(source, dest, vol_source, vol_dest=None)
unqueued()
```

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 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])

continues on next page
```

Table 11 – continued from previous page

lable 11 – continue	ed from previous page
<pre>blowOutCellLegacy([cellname])</pre>	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
cellToWaste([cellname])	
deposit_obj(obj[, uid])	Store an object in the dropbox
drySyringe([blow, waittime])	transfer from air to waste, to push out any residual liquid.
execute(**kwargs)	1
<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_execute(**kwargs)	Executed before each call to execute
queued()	Executed before each earl 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])	remere un coject nom the dropoon
rinseCatch()	
rinseCell([cellname])	
rinseCellFlood([cellname])	
rinseCellPull([cellname])	
rinseSyringe()	
setRinseLevel(vol)	
setWasteLevel(vol)	
	continues on next page

continues on next page

Table 11 – continued from previous page

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
     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)
```

AFL.automation.loading.OneSelectorBlowoutSampleCell.defaultdict

${\bf class} \ {\tt AFL.} automation. loading. One Selector {\tt BlowoutSampleCell.} {\bf default dict}$

```
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)
```

unqueued()

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.
<pre>setdefault(key[, default])</pre>	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().
ini. (*	
init(*args, **kwargs)	
$\textbf{clear()} \rightarrow None. \ Remove \ all \ items \ from \ D.$	
$copy() \rightarrow a$ 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 at	nd values set to value.
<pre>get(key, default=None, /)</pre>	
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 it	ems
$\mathbf{keys}() \to a$ set-like object providing a view on D's key	VS
$pop(k[,d]) \rightarrow v$, remove specified key and return the If the key is not found, return the default if given;	

```
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
                                                                                                                  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)
```

popitem()

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

 ${\bf class} \ \, {\tt AFL.automation.loading.PneumaticPressureSampleCell.Driver} (name, defaults=None, overrides=None)$

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
         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 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})

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>	
<pre>loadSample([cellname, sampleVolume,])</pre>	Load a sample into the cell
<pre>post_execute(**kwargs)</pre>	Executed after each call to execute
<pre>pre_execute(**kwargs)</pre>	Executed before each call to execute

continues on next page

Table 12 – continued from previous page

```
primeRinse([waittime])
queued()
quickbar()
read_sensor()
read_sensor_poll(**kwargs)
read_sensor_poll_load(**kwargs)
reset_sample()
reset_tank_levels([rinse1, rinse2, waste])
                                                 Retrieve an object from the dropbox
retrieve_obj(uid[, delete])
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])
\verb|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
         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(**kwargs)
read_sensor_poll_load(**kwargs)
set_sensor_config(**kwargs)
get_sensor_config(**kwargs)
sensor_reset(sensor_n=None)
```

AFL.automation.loading.PneumaticPressureSampleCell.SampleCell

```
\textbf{class} \ \texttt{AFL}. automation. loading. Pneumatic Pressure Sample Cell. \textbf{Sample Cell}
```

```
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)	
${f clear}() ightarrow {f N}{f one}.$ Remove all items from D.	
$copy() \rightarrow a \text{ shallow copy of D.}$	
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	y, else default.
$\textbf{items()} \rightarrow a \text{ set-like object providing a view on D's i}$	tems
$\textbf{keys}(\textbf{)} \rightarrow \textbf{a}$ set-like object providing a view on D's ke	eys
$pop(k[,d]) \rightarrow v$, remove specified key and return the If the key is not found, return the default if given	

```
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()
```

popitem()

```
{\bf loadSample} ({\it 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_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])
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()
                                                   Executed after each call to execute
post_execute(**kwargs)
pre_execute(**kwargs)
                                                   Executed before each call to execute
queued()
quickbar()
reset_sample()
                                                   Retrieve an object from the dropbox
retrieve_obj(uid[, delete])
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.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)

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})

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>	
queued()	

continues on next page

Table 13 – continued from previous page

```
quickbar()
read_sensor()
read_sensor_poll(**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

```
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
class AFL.automation.loading.PneumaticSampleCell.SampleCell
     Bases: object
     __init__()
```

```
__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.

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.
<pre>setdefault(key[, default])</pre>	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 by __missing__().
      __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.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

AFL.automation.loading.PressureController.PressureController

class AFL.automation.loading.PressureController.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.

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()
```

AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump

```
 \begin{tabular}{ll} {\bf class} & {\tt AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump}(pressure\_controller, \\ & dispense\_pressure=3.5, \\ & im-plied\_flow\_rate=50) \end{tabular}
```

```
Bases: SyringePump
__init__(pressure_controller, dispense_pressure=3.5, implied_flow_rate=50)
Initialize a pressure controller as a syringe pump.
```

 $pressure_controller: controller to connect to dispense_pressure: pressure at which dispense should run implied_flow_rate: flow rate which should be used to convert to dispense times, mL/min$

```
_init__(pressure_controller[, ...])
                                                       Initialize a pressure controller as a syringe pump.
 blockUntilStatusStopped([pollingdelay])
 dispense(volume[, block, delay])
 emptySyringe()
 getRate()
 getStatus()
                                                       query the pump status and return a tuple of the status
                                                       character, infused volume, and withdrawn volume)
 setRate(rate)
                                                       Abort the current dispense/withdraw action.
 stop()
 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

Bases: object

__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)

```
__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()
```

 $\textbf{class} \ \, \textbf{AFL}. \textbf{automation.} loading. \textbf{PressureControllerAsPump.} \textbf{\textit{PressureControllerAsPump.} PressureControllerAsPump.} \\ \textbf{\textit{dis-}}$

pense_pressure=3.5,
im-

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

AFL.automation.loading.PushPullSelectorSampleCell.Driver

```
__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
         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,
```

```
selec-
tor,
ncells=1,
thick-
ness=None,
catch to sel vol=None,
cell_to_sel_vol=None,
sv-
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)
```

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])
```

6.2. Modules 297

continues on next page

Table 14 – continued from previous page

Table 14 – Continue	d from previous page
<pre>blowOutCellForcedAir([cellname, waittime])</pre>	
<pre>catchToSyringe([sampleVolume])</pre>	
<pre>catchToWaste([sampleVolume])</pre>	
cellToWaste([cellname])	
deposit_obj(obj[, uid])	Store an object in the dropbox
execute(**kwargs)	
<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>	
post_execute(**kwargs)	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])	
rinseSyringe()	
setParameter(parameter, value)	
set_config(**kwargs)	
set_data(data)	Set data in the DataPacket object

continues on next page

Table 14 - continued from previous page

```
set_object([serialized])
set_sample(sample_name[, sample_uuid])
status()
swish(vol)
transfer(source, dest, vol_source[, vol_dest])
unqueued()
```

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
```

```
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.PushPullSelectorSampleCell.defaultdict

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.

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 by __missing__().
__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.PushPullSelectorSampleCell.PushPullSelectorSampleCell(pump,
```

```
selec-
tor,
ncells=1,
thick-
ness=None,
catch to sel vol=None,
cell_to_sel_vol=None,
sv-
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
     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(default_factory=None, /, [...]) --> dict with default factory
```

AFL.automation.loading.RSoXSSolutionSampleCell.Driver

```
__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)
```

cell_waste_tank_level=0,

over-

rides=None)

```
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.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell
class AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell(pump, selector,
                                                                                             rinse tank level=950,
                                                                                             waste\_tank\_level=0,
```

```
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

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>	
<pre>blowOutCellLegacy([cellname])</pre>	
<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.
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>	

continues on next page

Table 15 – continued from previous page

```
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])
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?
     Methods
       __init__(specid, length)
                                                      length is in cm?
                                                      returns volume in mL
      volume()
```

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
```

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)
```

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.
<pre>setdefault(key[, default])</pre>	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().
ini. (*	
init(*args, **kwargs)	
$\textbf{clear()} \rightarrow None. \ Remove \ all \ items \ from \ D.$	
$copy() \rightarrow a$ 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 at	nd values set to value.
<pre>get(key, default=None, /)</pre>	
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 it	ems
$\textbf{keys}() \rightarrow a \text{ set-like object providing a view on D's keys}$	
$pop(k[,d]) \rightarrow v$, 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.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

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

```
MultiChannelRelay()

SainSmartRelay(relaylabels, serial_port[, ...])

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

AFL.automation.loading.SainSmartRelay.MultiChannelRelay

while the 1st index selects the array row.

```
__init__()

getChannels(channels)

setChannels(channels)

toggleChannels(channels)

setChannels(channels)

getChannels(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

```
__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':FE05000FFF00FF\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)
     667777
```

AFL.automation.loading.SampleCell

Classes

320

```
SampleCell()
```

AFL.automation.loading.SampleCell.SampleCell

```
class AFL.automation.loading.SampleCell.SampleCell
    Bases: object
```

```
__init__()
```

```
__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

```
\textbf{class} \ \texttt{AFL.automation.loading.Sensor.DummySensor1} (\textit{period} = 2, \textit{minval} = -5, \textit{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.

 ${\it args}$ 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.

Methods

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
```

6.2. Modules 327

the args and kwargs arguments, respectively.

to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from

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.

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

```
\textbf{class} \  \, \textbf{AFL}. \textbf{automation.loading.SensorCallbackThread.} \\ \textbf{LoaderCommunication} (load\_client=None, \\ load\_object=None)
```

```
Bases: object
__init__(load_client=None, load_object=None)
```

```
__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.} 1 \textbf{o} \textbf{a} \textbf{d} \textbf{i} \textbf{g}. \textbf{SensorCallbackThread}. \textbf{SensorCallbackThread} (\textit{poll}, \textit{period} = 0.1, \\ \textit{d} \textbf{a} \textbf{e} \textit{mon} = \textit{True}, \\ \textit{filepath} = \textit{None}, \\ \textit{d} \textbf{a} \textbf{t} \textbf{a} = \textit{None}, \\ \textit{sensorlabel} = ")
```

Bases: Thread

 $\verb|__init__(poll, period=0.1, daemon=True, filepath=None, data=None, sensor label=")|$

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, 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.SensorCallbackThread.SimpleThresholdCB

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.

Methods

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.

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.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.

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)
```

 $\textbf{class AFL.automation.loading.SensorCallbackThread.SensorCallbackThread} (poll, period = 0.1, \\ daemon = True, \\ filepath = None, \\ data = None, \\ sensorlabel = ")$

__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

 $\textbf{class} \ \texttt{AFL}. automation. loading. Sensor Polling Thread. \textbf{Sensor Polling Thread} (\textit{sensor}, \textit{period} = 0.1, \texttt{automation}) and the property of the$

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.

 $\ensuremath{\mathit{args}}$ 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(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().

ioin(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.

```
\textbf{class} \  \, \textbf{AFL.automation.loading.SensorPollingThread}. \textbf{SensorPollingThread} (\textit{sensor}, \textit{period} = 0.1, \\ \textbf{sensorPollingThread}). \\ \textbf{sensorPollingThread} (\textit{sensor}, \textit{period} = 0.1, \\ \textbf{sensor}, \textit{period}
```

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, ...])
```

AFL.automation.loading.SerialDevice.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)

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)

```
__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
               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__()
```

```
__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

```
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. Ioading. Two Selector Blowout Sample Cell. Two Selector Blow Sele

```
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)
```

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
<pre>blowOutCell([cellname, waittime])</pre>	
blowOutCellLegacy([cellname])	
<pre>catchToSyringe([sampleVolume])</pre>	
catchToWaste([sampleVolume])	
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

continues on next page

Table 16 – 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 16 - continued from previous page

```
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
     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()
```

AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultdict

class AFL.automation.loading.TwoSelectorBlowoutSampleCell.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()	
<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 bymissin	g().	
<pre>fromkeys(value=None,/)</pre>		
Create a new dictionary with keys from itera	able and values set to value.	
<pre>get(key, default=None, /)</pre>	1 16 1	
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 I	D's keys	
$\mathtt{pop}(k \llbracket, d \rrbracket) \to \mathrm{v}$, remove specified key and return	rn the corresponding value.	
If the key is not found, return the default if g	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-
                                                                                                         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)
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, pense_time)</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

class AFL.automation.loading.UltimusVPressureController.UltimusVPressureController(port,

baud=115200)

Methods

init(port[, baud]) blockUntilStatusStopped([pollingdelay])		Initializes a DigitalOutPressureController block execution until the controller finishes a dispense
char_count(cmd)		
<pre>compute_checksum(cmd)</pre>		
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.

```
__init__(port, baud=115200)
```

Initializes a DigitalOutPressureController

Params:

port (str): serial port to use

 ${\tt compute_checksum}(cmd)$

char_count(cmd)

package_cmd(cmd)

send_command(cmd)

set_P(pressure)

pressure: pressure to set in psi

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().

 $\textbf{class} \ \texttt{AFL}. automation. loading. Ultimus VP ressure \texttt{Controller}. \textbf{Ultimus VP ressure \texttt{Controller}} (port, port, p$

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

```
{\bf class} \ {\tt AFL.automation.loading.ViciMultiposSelector.} {\bf 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

- 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

- 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__()

```
__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_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()
 {\tt 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])
 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()
process_components()
```

```
make_mass_fraction_matrix()
     make_target_component_masses()
     balance_mass()
     sample_composition_space(pipette_min=<Quantity(5, 'microliter')>,
                                   stock_transfer_max=<Quantity(3500, 'microliter')>, grid_density=5,
                                   composition_data='frac')
          Combine stock solutions to generate samples of possible target compositions
     constrain_samples_conc(constraints, rtol=0.05)
     calculate_bounds(components=None, exclude_comps_below=None, fixed_comps=None,
                         composition_data='samples_frac', ternary=True)
     in_bounds(points, ternary=True)
     make_grid_mask(pts_per_row=100, ternary=True)
     plot_grid_mask()
     plot_bounds(include_points=True)
AFL.automation.prepare.PipetteAction
class AFL.automation.prepare.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()
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
```

6.2. Modules 373

__init__()

Methods

```
__init__()
add_sample(sample)

mass_totals_component([only_validated])

reset()
shuffle()

__init__()
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
property is_solvent
property mass
property moles
set_mass(value)
    Setter for inline mass changes
set_volume(volume)
    Setter for inline volume changes
```

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[,
                                      properties,
...])
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[, in-
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 of components in mixture
mass_fraction
molarity
size
solutes
solvent_density
solvent_mass
solvent_sld
solvent_volume
solvents
                                                Total volume of mixture.
volume
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
                  • 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

fault_units	<pre>enforce_units(value, unit_type)</pre>	Ensure that a number has units and convert to the default_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__()
```

6.2. Modules 381

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
class AFL.automation.prepare.Component.PrepType(value, names=None, *, module=None,
                                                   qualname=None, type=None, start=1,
                                                   boundary=None)
     Bases: Enum
     __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:

emit()

```
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)
```

```
__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, /)

384

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()	
DeckBuilderWidget_Model()	
<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.
7 1 7/441	
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.
	continues on post page

continues on next page

Table 17 – continued 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>	
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_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 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 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)	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 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) → 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) \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) \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.

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.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.
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.

continues on next page

Table 18 – continued from previous page

	han han branche baile
<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.
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.
<pre>setup_instance(**kwargs)</pre>	This is called before 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.
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

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) \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.

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.

continues on next page

- 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])

enqueued_base(**kwargs)

from_server_name(server_name, **kwargs)

get_config(name[, print_console, interactive])
```

Table 19 – continued from previous page

```
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_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.DeckBuilderWidget.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.DeckBuilderWidget.DeckBuilderWidget_Model

```
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_Model
    Bases: object
    __init__()
```

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
class AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget_View
     Bases: object
     __init__()
     Methods
      __init__()
      create_expanded_button(description,
                                               but-
      ton 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
```

[('Galileo', 0), ('Brahe', 1), ('Hubble', 2)], or a Mapping between labels and values, e.g., {'Galileo': 0, 'Brahe': 1, 'Hubble': 2}.

• **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.

- 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 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.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
	continues on payt page

continues on next page

Table 20 – 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 before 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) → 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) \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

keys

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

410

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

```
class AFL.automation.prepare.DeckBuilderWidget.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])

__init__(children=(), **kwargs)
```

Public constructor

Methods

	D 11:
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.
<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 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.
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 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.
<u></u>	continues on next page

continues on next page

Table 21 – continued from previous page

<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) \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.

$\verb|static get_manager_state|| \textit{drop_defaults} = False, \textit{widgets} = None||$

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:

- \ast owner: the HasTraits instance \ast old: the old value of the modified trait attribute \ast new: the new value of the modified trait attribute \ast 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 | <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

(5.1.1)	D 111
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.
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.
	continues on next page

continues on next page

Table 22 – continued from previous page

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.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called before 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.
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) $\rightarrow list[str]$

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 = '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) \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 = '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)
```

Layout specification

Bases: Widget

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

L L Child	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>	
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.
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 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 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)	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.
·	continues on next ness

continues on next page

Table 23 – continued from previous page

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.
<pre>grid_template_areas</pre>	The grid-template-areas CSS attribute.
<pre>grid_template_columns</pre>	The grid-template-columns CSS attribute.
<pre>grid_template_rows</pre>	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.
overflow	The overflow CSS attribute.
padding	The padding CSS attribute.

continues on next page

Table 24 – continued from previous page

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

$hold_trait_notifications() \rightarrow Any$

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.

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 \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) \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.

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:
```

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

```
class AFL.automation.prepare.DeckBuilderWidget.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
has_trait(name)	Returns True if the object has a trait with the specified name.

continues on next page

Table 25 – continued from previous page

	, , ,
hold_sync()	Hold syncing any state until the outermost context manager exits
hold_trait_notifications()	Context manager for bundling trait change notifica-
nord_trart_notrifications()	tions and cross validation.
notify_change(change)	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.
send(content[, buffers])	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.
<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 before 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.
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

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

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) \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.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])
```

```
__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 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.
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.

continues on next page

Table 26 – continued from previous page

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 before 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() → 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.
     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])
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()
                                                   Executed after each call to execute
post_execute(**kwargs)
                                                   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()
```

```
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])	
<pre>add_prep_targets(targets[, reset])</pre>	
<pre>deactivate_temp(slot)</pre>	Disablea tempdeck in slot slot
deposit_obj(obj[, uid])	Store an object in the dropbox
execute(**kwargs)	Store an object in the dropoox
execute(~kwaigs)	
<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>	
get_conrigo([print_console])	
<pre>get_labware(slot)</pre>	
get_labware(slot)	
. 1 / [1 .])	
<pre>get_object(name[, serialize])</pre>	
<pre>get_prep_target()</pre>	
<pre>get_sample()</pre>	
<pre>get_shake_latch_status()</pre>	
<pre>get_shake_rpm()</pre>	
<pre>get_shaker_temp()</pre>	
ge e_brianci _ temp()	
<pre>get_temp(slot)</pre>	Get the temperature of a tempdeck in slot slot
	Get the temperature of a tempacek in slot slot
<pre>get_wells(locs)</pre>	
I doubt	
home(**kwargs)	
latch_shaker()	
<pre>load_instrument(name, mount, tip_rack_slots,)</pre>	
<pre>load_labware(name, slot[, module])</pre>	
parse_well(loc)	
par be_werr(ioc)	
post_execute(**kwargs)	Executed after each call to execute
- · · · · · · · · · · · · · · · · · · ·	Executed after each call to execute
pre_execute(**kwargs)	
	continues on next page

continues on next page

Table 27 – continued from previous page

```
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])
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()
```

continues on next page

Table 28 – continued from previous page

```
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)
 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()
get_unqueued_commands([inherit_commands])
```

continues on next page

Table 29 - continued from previous page

```
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.

      PrepType(value[, names, module, qualname, ...])
      Instances are replaced with an appropriate value in Enum class suites.
```

AFL.automation.prepare.PrepType.Enum

class AFL.automation.prepare.PrepType.**Enum**(*value*, *names=None*, *, *module=None*, *qualname=None*, type=None, start=1, boundary=None)

Bases: object

Create a collection of name/value pairs.

Example enumeration:

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

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

```
Bases: Enum
__init__(*args, **kwds)
```

Attributes

```
BaseComponent

BaseMixture

Solute

Solvent

Solution
```

```
BaseMixture = 2
Solute = 3
Solvent = 4
Solution = 5
```

BaseComponent = 1

```
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()	1 0
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.

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

add_class(className)Adds a class to the top level element ofadd_traits(**traits)Dynamically add trait attributes to the V	
add_traits(**traits) Dynamically add trait attributes to the V	he widget.
	/idget.
blur() Blur the widget.	

continues on next page

Table 30 – continued from previous page

Table 30 – continu	ed from previous page
<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.
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.
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.
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.
<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.
·	continues on next nage

continues on next page

Table 30 – continued from previous page

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

$hold_trait_notifications() \rightarrow Any$

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.

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.

```
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

- $\bullet \ \ \textbf{value} \ (\{\textit{True}, \textit{False}\}) \text{value of the checkbox: True-checked}, \\ \text{False-unchecked}$
- **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.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.

continues on next page

Table 31 – continued from previous page

Table 31 – continue	1 1 0
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
close_all()	
5	-
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-
and add (flow days defaulted)	ding
<pre>get_state([key, drop_defaults]) get_view_spec()</pre>	Gets the widget state, or a piece of it.
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 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 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 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 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
<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.
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.
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) \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.

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

```
class AFL.automation.prepare.PrepareWidget.DeckBuilderWidget
```

```
Bases: object
__init__()
```

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

```
class AFL.automation.prepare.PrepareWidget.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

	w 11.
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 32 – continued from previous page

get_interact_value() Return the value for this widget which should be passed to interactive functions. 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_control_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]) Called when a property has changed. on_msg(callback], remove) Called when a property has changed. on_msidget_constructed(callback) DEPRECATED: Setup a handler to be called when a trait changes. on_widget_constructed(callback) Registers a custom msg receive callback. open() Depart of the constructed.	Table 32 – continue	d Iron previous page
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_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 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]) Called when a property has changed. on_msg(callback[, remove]) Cun)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 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.	<pre>get_interact_value()</pre>	Return the value for this widget which should be
ding Gets the widget state, or a piece of it. 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 context manager exits hold_trait_notifications() collect 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) on_widget_constructed(callback) send_state([key]) send_st		passed to interactive functions.
get_state([key, drop_defaults]) Gets the widget state, or a piece of it. get_view_spec() handle_comm_opened(comm, msg) 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 has_trait(name) 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(content[, buffers]) Sends a custom msg to the widget model in the frontend. <tr< td=""><td><pre>get_manager_state([drop_defaults, widgets])</pre></td><td></td></tr<>	<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 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. called when a property has changed. Setup a handler to be called when a trait changes. On_msg(callback , remove) Culled when a property has changed. 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. Sends a custom msg to the widget model in the frontend. set_state([key]) Sends he widget state, or a piece of it, to the frontend, if it exists. Set_state(sync_data) Set_state(sync_data) 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. 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' traits. Trait_metadata(traitname, key[, default]) Get a dict of all the names of this class' traits. Trait_alexes(**metadata) Get a dict of trait names and their values. Traits(**metadata) Get a dict of trait names and their values. Traits(**metadata) Get a dict of trait names and their values. Traits(**metadata) Get a dict of trait names and their values. Traits(**metadata) Get a dict of trait names and their values. Traits(**metadata)		
handle_comm_opened(comm, msg) 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 Context manager for bundling trait change notifications and cross validation. notify_change(change) Observe(handler[, names, type]) On_msg(callback[, remove]) Called when a property has changed. on_msg(callback[, remove]) On_trait_change([handler, name, remove]) on_widget_constructed(callback) notify_change(change) Called when a property has changed. DEPRECATED: Setup a handler to be called when a trait changes. On_widget_constructed(callback) Register a custom msg receive callback. 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_state([key]) Sends a custom msg to the widget model in the frontend. set_state(sync_data) Set_rait(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_has_value(name) Returns True if the opication on the med, and their values. Trait_values(**metadata) Cet a dict of all the raits of this class. Traits of this class. Cet a dict of all the traits of this class.		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) 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_sadult([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 frontend, if it exists. setup_instance(**kwargs) This is called before selfinit is called. trait_defaults(*names, **metadata) Return a trait's default value or a dictionar	<pre>get_view_spec()</pre>	
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) 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_sadult([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 frontend, if it exists. setup_instance(**kwargs) This is called before selfinit is called. trait_defaults(*names, **metadata) Return a trait's default value or a dictionar		
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]) on_msg(callback[, remove]) on_trait_change([handler, name, remove]) on_trait_change([handler, name, remove]) on_widget_constructed(callback) pen() pen() pen 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. called when a state is received from the frontend. set_state(sync_data) set_state(sync_data) set_rait(name, value) Forcibly sets trait attribute, including read-only attributes. setup_instance(**kwargs) trait_defaults(*names, **metadata) trait_events([name]) trait_names(**kmargs) trait_names(**metadata) trait_names(**metadata) trait_names(**metadata) Cet a dict of all the event handlers of this class. trait_values(**metadata) A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.		
ceived 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. 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. set_trait(name, value) Forcibly sets trait attribute, including read-only attributes. setup_instance(**kwargs) trait_defaults(*names, **metadata) trait_events([name]) Get a dict of all the event handlers of this class. trait_names(**metadata) trait_values(**metadata) trait_values(**metadata) Cet a dict of all the names of this class. trait_values(**metadata) Cet a dict of all the names of this class.	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 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_trait_change([handler, name, remove]) 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. 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 Get a dict of all the event handlers of this class. <t< td=""><td></td><td>• • •</td></t<>		• • •
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_sate([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. 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(*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) A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.		
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.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_has_value(name)Returns True if the specified trait has a value.trait_mames(**metadata)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.traits(**metadata)A dict of trait names and their values.traits(**metadata)Get a dict of all the tr	has_trait(name)	
manager exits Context manager for bundling trait change notifications and cross validation. notify_change(change)	1 11 0	
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(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_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_newts([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.traits(**metadata)A dict of trait names and their values.traits(**metadata)Get a dict of a	no1a_sync()	
tions and cross validation. Called when a property has changed. Observe(handler[, names, type]) Ommsg(callback[, remove]) Ommsg(callback[, remove]) Ommtait_change([handler, name, remove]) Om_trait_change([handler, name, remove]) Om_widget_constructed(callback) Omen() 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, if it exists. Set_state([key]) Sends the widget state, or a piece of it, to the frontend, if it exists. Set_trait(name, value) Set_trait(name, value) This is called before selfinit is called. trait_defaults(*names, **metadata) trait_metadata(traitname, key[, default]) trait_names(**metadata) trait_names(**metadata) trait_names(**metadata) trait_names(**metadata) Called when a property has changed. Called when a property has changed. Setup a handler to be called when a trait changes. Outpressed 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 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) 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. This is called before trait by key. Get metadata values for trait by key. Get metadata values for trait by key. Get a list of all the names of this class' traits. Thait_values(**metadata) A dict of trait names and their values. Get a dict of all the traits of this class.	hald trait matifications()	
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, if it exists.set_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_names(**metadata(traitname, key[, default])Returns True if the specified trait has a value.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 names of this class' traits.traits(**metadata)A dict of trait names and their values.traits(**metadata)Get a dict of all the traits of this class.	noid_trait_notifications()	
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 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_has_value(name)Returns True if the specified trait has a value.trait_metadata(traitname, key[, default])Get a dict of all the event handlers of this class' traits.trait_values(**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.	notify shange(shange)	
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 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_has_value(name)Returns True if the specified trait has a value.trait_metadata(traitname, key[, default])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)A dict of trait names and their values.traits(**metadata)Get a dict of all the traits of this class.		
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_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.traits(**metadata)A dict of trait names and their values.traits(**metadata)Get a dict of all the traits of this class.		•
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. 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_events([name]) Get a dict of all the event handlers of this class. trait_metadata(traitname, key[, default]) 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.		
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_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.	on_trart_change([nandler, name, remove])	
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 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_events([name]) Get a dict of all the event handlers of this class. trait_metadata(traitname, key[, default]) 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. trait_values(**metadata) Get a dict of all the traits of this class.	on widget constructed(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(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.	on_wruget_constructed(canback)	<u> </u>
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.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.	onen()	
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) 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) A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.		•
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.	Temove_erass(erassivarite)	*
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) A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.	send(content[, buffers])	e e
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) Setup_instance(**kwargs) 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]) 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.	content, content)	
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]) Cet 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.	send_state([kev])	
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_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.	_ (L) J/	
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) Get a dict of all the event handlers of this class. Get metadata values for trait by key. 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.	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_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.		Forcibly sets trait attribute, including read-only at-
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_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.		
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. A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.	setup_instance(**kwargs)	This is called before selfinit is called.
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.		
trait_metadata(traitname, key[, default]) trait_names(**metadata) Get metadata values for trait by key. 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.		
trait_names(**metadata) trait_values(**metadata) traits(**metadata) 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.		Returns True if the specified trait has a value.
trait_values(**metadata) A dict of trait names and their values. traits(**metadata) Get a dict of all the traits of this class.		
traits(**metadata) Get a dict of all the traits of this class.		
Domaria a trait abanga bandlar		
	<pre>unobserve(handler[, names, type])</pre>	Remove a trait change handler.
unobserve_all([name]) Remove trait change handlers of any type for the spec-	unobserve_all([name])	• • • • • • • • • • • • • • • • • • • •
ified name.		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
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) → 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() → 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.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

	7.11
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.
<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>	
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.
<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 at-
	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)	Returns True if the specified trait has a value.
,	continues on next page

continues on next page

Table 33 – continued from previous page

<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) \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) \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.

focus()

Focus on the widget.

$\verb|static get_manager_state|| \textit{drop_defaults} = False, \textit{widgets} = None||$

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:

- \ast owner: the HasTraits instance \ast old: the old value of the modified trait attribute \ast new: the new value of the modified trait attribute \ast 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) \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.

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>	
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.
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.
- "	continues on next page

continues on next page

Table 34 – continued from previous page

7 (1)	
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.
set_trait(name, value)	Forcibly sets trait attribute, including read-only at-
	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)	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.
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) \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 = '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) \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 = '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

init(**kwargs)	Public constructor
add_traits(**traits)	Dynamically add trait attributes to 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()	
<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
<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.
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.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only at-
	tributes.
setup_instance(**kwargs)	This is called before 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.
	continues on next name

continues on next page

Table 35 – continued from previous page

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

alian content	The clien content CCC ettribute
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
109	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.
overflow	The overflow CSS attribute.
padding	The padding CSS attribute.
padding	continues on port page

continues on next page

Table 36 – continued from previous page

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

$hold_trait_notifications() \rightarrow Any$

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.

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.

```
\begin{tabular}{ll} \textbf{on\_trait\_change}(handler: EventHandler \mid None = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, name: Sentinel \mid str \mid None = None, remove: bool \\ = False) \rightarrow \begin{tabular}{ll} \textbf{None} = None, remove:
```

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 \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) \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.

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:
```

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

```
class AFL.automation.prepare.PrepareWidget.PrepareWidget
    Bases: object
    __init__()
```

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

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
class AFL.automation.prepare.PrepareWidget.StockBuilderWidget(deck)
     Bases: object
     __init__(deck)
     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])
```

6.2. Modules 527

set_units_cb(event, units)

update_location_check_cb(event)

start()

```
__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
class AFL.automation.prepare.PrepareWidget.SweepBuilderWidget(deck)
     Bases: object
     __init__(deck)
     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
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
	continues on next page

Table 37 – continued from previous page

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.
<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.
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 before 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.
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

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

 $\ensuremath{\textit{key}}\xspace(unicode\ or\ iterable\ (optional))$ — 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) \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.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])
```

```
__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 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.
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.

continues on next page

Table 38 – continued from previous page

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 before 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() → 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 | str |
```

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)
```

class AFL.automation.prepare.PrepareWidget.PrepareWidget_Model

class AFL.automation.prepare.PrepareWidget.PrepareWidget_View

start(deck_builder_widget)

AFL.automation.prepare.SampleSeriesWidget

Functions

start()

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 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.

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.
<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.

continues on next page

Table 39 – continued from previous page

	d from previous page
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 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
<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_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 front-
	end.
<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 before 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

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 | 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) \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.

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:
```

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.Checkbox

```
class AFL.automation.prepare.SampleSeriesWidget.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.
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.
<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.

continues on next page

Table 40 – continued from previous page

<pre>get_view_spec()</pre>	
handle_comm_opened(comm, msg)	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
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_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 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)	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
1-1 -2 1	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) \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.

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:
```

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 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])

enqueued_base(**kwargs)

from_server_name(server_name, **kwargs)

get_config(name[, print_console, interactive])
```

continues on next page

Table 41 – continued from previous page

```
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_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)
```

6.2. Modules 563

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

class AFL.automation.prepare.SampleSeriesWidget.FloatText(**kwargs: Any)

Bases: _Float

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

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.
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.
<pre>on_msg(callback[, remove])</pre>	(Un)Register a custom msg receive callback.
	continues on next page

continues on next page

Table 42 – continued from previous page

	1 1 0
<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 before 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.
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

 $\begin{tabular}{ll} \textbf{key} (unicode \ or \ iterable \ (optional)) - A single property's name or iterable of property names to get. \\ \end{tabular}$

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 \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) → 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 = '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])
```

```
__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.
<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

continues on next page

Table 43 – continued from previous page

	1 0
<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.
<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.
<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.
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 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

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) \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

$hold_trait_notifications() \rightarrow Any$

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) \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.

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.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.
<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.
class_traits(**metadata)	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 44 – 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 frontend, if it exists.
Called when a state is received from the front-end.
Forcibly sets trait attribute, including read-only attributes.
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.
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.
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.

$\textbf{classmethod class_traits(**metadata: Any)} \rightarrow \text{dict[str, TraitType[Any, Any]]}$

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.

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)
```

Methods

Public constructor

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.
class_traits(**metadata)	Get a dict of all the traits of this class.

continues on next page

Table 45 – continued from previous page

Table 40 Continue	ed from previous page
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
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
and (content [1 (Cont)	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-
ant atata(aura data)	end, if it exists. Called when a state is received from the front-end.
set_state(sync_data)	
set_trait(name, value)	Forcibly sets trait attribute, including read-only at-
cotun instance(**kweese)	tributes. This is called before calf init is called
<pre>setup_instance(**kwargs) trait_defaults(*names, **metadata)</pre>	This is called before selfinit is called.
trait_defaults(*names, **metadata) trait_events([name])	Return a trait's default value or a dictionary of them Get a dict of all the event handlers of this class.
trait_events([name]) trait_has_value(name)	Returns True if the specified trait has a value.
trait_nas_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-
anobserve_arr([name])	ified name.
	med 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) \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() → 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) \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.

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:
```

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

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.
<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.
	continues on next nage

continues on next page

Table 46 – continued from previous page

	ou nom promote 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
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.
<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.
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.
<pre>setup_instance(**kwargs)</pre>	This is called before 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.
<pre>unobserve_all([name])</pre>	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.

continues on next page

Table 47 – continued from previous page

The flex CSS attribute.
The flex-flow CSS attribute.
The grid-area CSS attribute.
The grid-auto-columns CSS attribute.
The grid-auto-flow CSS attribute.
The grid-auto-rows CSS attribute.
The grid-column CSS attribute.
The grid-gap CSS attribute.
The grid-row CSS attribute.
The grid-template-areas CSS attribute.
The grid-template-columns CSS attribute.
The grid-template-rows CSS attribute.
The height CSS attribute.
The justify-content CSS attribute.
The justify-items CSS attribute.
The traits which are synced.
The left CSS attribute.
A trait whose value must be an instance of a specified
class.
The margin CSS attribute.
The max-height CSS attribute.
The max-width CSS attribute.
The min-height CSS attribute.
The min-width CSS attribute.
Gets the model id of this widget.
The object-fit CSS attribute.
The object-position CSS attribute.
The order CSS attribute.
The overflow CSS attribute.
The padding CSS attribute.
The right CSS attribute.
The top CSS attribute.
The visibility CSS attribute.
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) \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.

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

 $\begin{tabular}{ll} \textbf{key} (unicode \ or \ iterable \ (optional)) - A single property's name or iterable of property names to get. \\ \end{tabular}$

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.

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.widget.WidgetRegistry object>
widgets = {}
```

AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget

```
class AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget(deck)
    Bases: object
    __init__(deck)
```

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
\textbf{class} \ AFL. automation. prepare. Sample Series \texttt{Widget}. \textbf{Sample Series} \textbf{Widget}\_\textbf{Model} (\textit{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

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

continues on next page

Table 48 – continued from previous page

	a nom previous page
<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
<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.
<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.
<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
1/	widget.
send(content[, buffers])	Sends a custom msg to the widget model in the front-
(7)	end.
send_state([key])	Sends the widget state, or a piece of it, to the front-
1.1	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 before selfinit is called.
<pre>trait_defaults(*names, **metadata) trait_events([name])</pre>	Return a trait's default value or a dictionary of them Get a dict of all the event handlers of this class.
trait_events([name]) trait_has_value(name)	
	Returns True if the specified trait has a value.
<pre>trait_metadata(traitname, key[, default]) trait_names(**metadata)</pre>	Get metadata values for trait by key. 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-
anobserve_arr([name])	ified name.
	med 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) \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) \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 = '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 \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 = '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.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])
```

```
__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 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.
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.

continues on next page

Table 49 – continued from previous page

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.
<pre>setup_instance(**kwargs)</pre>	This is called before 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 el
```

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.

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. 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) 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, /)

Return the square root of x.

AFL.automation.prepare.StockBuilderWidget.sqrt

AFL.automation.prepare.StockBuilderWidget. $\mathbf{sqrt}(x,/)$ Return the square root of x.

Classes

```
StockBuilderWidget(deck)
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

```
__init__(deck)
```

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)
to_stock_objects(all_stocks_dict)
```

AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget_View

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.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, l)	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.
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.

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.
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 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.
	continues on next page

continues on next page

Table 50 – continued from previous page

	a nom providuo pago
<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.
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.
<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) \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

 $\begin{tabular}{ll} \textbf{key} (unicode \ or \ iterable \ (optional)) - A single property's name or iterable of property names to get. \\ \end{tabular}$

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 \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) \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.

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.

```
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 ([valua])	Public constructor
init([value]) add_class(className)	Adds a class to the top level element of the widget.
	1
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 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 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.
	continues on next page

continues on next page

Table 51 – continued from previous page

	a nom providuo pago
<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 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 before 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.
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
```

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) \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.

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

- 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\_\_(\mathit{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.
<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>	
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.
<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.

continues on next page

Table 52 – continued from previous page

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.
<pre>set_trait(name, value)</pre>	Forcibly sets trait attribute, including read-only attributes.
<pre>setup_instance(**kwargs)</pre>	This is called before 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) \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) \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) \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.

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.SweepBuilderWidget.Label

```
class AFL.automation.prepare.SweepBuilderWidget.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.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
· /	Blur the widget.
blur()	ē
<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 embed-
	ding
<pre>get_state([key, drop_defaults])</pre>	Gets the widget state, or a piece of it.
<pre>get_view_spec()</pre>	<i>y</i> 1
• •	
handle_comm_opened(comm, msg)	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
	continues on next ness

continues on next page

Table 53 – 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_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.
<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 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)	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 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) \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() → 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) \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.

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
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.
	continues on poyt page

continues on next page

Table 54 – 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
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.
<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.
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.
<pre>setup_instance(**kwargs)</pre>	This is called before 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.
<pre>unobserve_all([name])</pre>	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.

continues on next page

Table 55 – continued from previous page

	Table 55 – continued from previous page
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.
<pre>grid_template_areas</pre>	The grid-template-areas CSS attribute.
<pre>grid_template_columns</pre>	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.
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) \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.

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.

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.

```
\textbf{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) \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.

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 | str |
```

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

```
class AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWidget(deck)
    Bases: object
    __init__(deck)
```

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. Sweep Builder Widget. Sweep Builder Widget\_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

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

```
class AFL.automation.prepare.SweepBuilderWidget.Text(**kwargs: Any)
    Bases: _String
    Single line textbox widget.
    __init__(*args, **kwargs)
    Public constructor
```

Methods

init(*args, **kwargs)	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.

continues on next page

Table 56 – continued from previous page

Table 56 – continued from previous page		
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.	
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.	
<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.	
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 before 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.	
	continues on next nage	

continues on next page

Table 56 – continued from previous page

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

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) \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 = '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])
```

```
__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 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.
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.

continues on next page

Table 57 – continued from previous page

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 before 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.
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*) → 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.

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. 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.SweepBuilderWidget_Model(deck) __init__(deck) validate_sweep(tolerance, progress=None) calc_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)

6.2. Modules 687

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.is_concentration(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

```
class AFL.automation.prepare.factory.Solution(name, components, properties=None)
    Bases: object
    __init__(name, components, properties=None)
```

Methods

```
__init__(name, components[, properties])
add_component_from_name(name[,
                                      properties,
...])
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 of components in mixture
mass_fraction
molarity
size
solutes
solvent_density
solvent_mass
solvent_sld
solvent_volume
solvents
                                                 Total volume of mixture.
volume
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.
```

Chapter 6. Reference

```
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.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, 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]) Driver(name[, defaults, overrides])</pre>	Communicate with APIServer
<pre>0T2Client([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])	
<pre>assign_targets(protocols, target_map)</pre>	
<pre>bulk_cast_films(**spec) deposit_obj(obj[, uid]) execute(**kwargs)</pre>	Spec should contain sample_name and a protocol Store an object in the dropbox
<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>init_casting_manifest([attrs, overwrite])</pre>	
<pre>log_event(manifest_key, event[, write])</pre>	
<pre>post_execute(**kwargs) pre_execute(**kwargs) prepare_and_cast(**sample)</pre>	Executed after each call to execute Executed before each call to execute
<pre>prepare_casting_stocks(**spec) queued()</pre>	Spec should contain sample_name and a protocol
quickbar()	
reset_sample()	
<pre>retrieve_obj(uid[, delete]) set_config(**kwargs)</pre>	Retrieve an object from the dropbox
set_data(data)	Set data in the DataPacket object
set_object([serialized])	Set data in the Datai acket object
<pre>set_sample(sample_name[, sample_uuid])</pre>	
status()	
unqueued()	
update_status(value)	

Attributes

```
defaults
```

```
defaults = {'manifest': '/home/afl642/', 'manifest_cast': '/home/afl642/',
'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)
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
                                                           interactive=False)
```

class AFL.automation.sample.CastingServer.**Client**(*ip=None*, *port='5000'*, *username=None*,

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 58 – continued from previous page

Table 36 – Continue	a nom providuo pago
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
driver_status()	1
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)	
<pre>queue_state()</pre>	
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
<pre>server_cmd(cmd, **kwargs)</pre>	, c
	continues on next page

continues on next page

Table 58 – 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)
```

```
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])
 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
               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
class AFL.automation.sample.CastingServer.0T2Client(ip=None, port='5000', username=None,
                                                               interactive=False)
     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()
get_unqueued_commands([inherit_commands])
halt()
home()
load_instrument(name, mount, tip_rack_slots)
load_labware(name, slot)
logged_in()
login(username[, populate_commands])
```

Table 59 - continued from previous page

```
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])
```

AFL.automation.sample_env.TemperatureDeck.Driver

```
class AFL.automation.sample_env.TemperatureDeck.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 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()
                                                   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 in the DataPacket object
set_data(data)
set_object([serialized])
set_sample(sample_name[, sample_uuid])
set_temp(sp)
status()
unqueued()
```

Attributes

quickbar()

reset_sample()

```
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()
```

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

AFL.automation.shared.DataLabelerWidget.sqrt

AFL.automation.shared.DataLabelerWidget. $\mathbf{sqrt}(x,/)$ Return the square root of x.

Classes

```
DataLabelerWodel(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()
```

AFL.automation.shared.DataLabelerWidget.DataLabelerView

get_peaks(model, qstar=None, max_order=4)

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)
```

```
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(data)

change_qstar_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()
```

```
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 $un-known_value$ 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.

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.
inverse_transform(X)	Convert the data back to the original representation.
<pre>set_output(*[, transform])</pre>	Set output container.
set_params(**params)	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.

Daturna

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)
rum()

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

722

AFL.automation.shared.DatasetWidget.DatasetWidget

- **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
- 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

```
class AFL.automation.shared.DatasetWidget.DatasetWidget_Model(dataset: Dataset, sample_dim: str)
    Bases: object
    __init__(dataset: Dataset, sample_dim: str)
```

Methods

```
__init__(dataset, sample_dim)

apply_isel(kw)

apply_sel(kw)

combine_vars(combined_var, to_combine_vars)

extract_var(extract_from_var, ex-
tract_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

```
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)
     Bases: object
```

__init__(initial_scatt_variables: List[str] | None = None, initial_comps_variable: str | None = None, initial_comps_color_variable: str | None = None)

Methods

```
__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, /, [...]) -> 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()	
<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.
<pre>setdefault(key[, default])</pre>	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().
ini. (*	
init(*args, **kwargs)	
$\textbf{clear()} \rightarrow None. \ Remove \ all \ items \ from \ D.$	
$copy() \rightarrow a$ 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 at	nd values set to value.
<pre>get(key, default=None, /)</pre>	
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 it	ems
$\mathbf{keys}() \to a$ set-like object providing a view on D's key	VS
$pop(k[,d]) \rightarrow v$, remove specified key and return the If the key is not found, return the default if given;	

```
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

```
__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.

xmin: float = 0.001, xmax: float = 1.0)

- **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)
     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

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

AFL.automation.shared.DiffractionLabeler.sqrt

AFL.automation.shared.DiffractionLabeler.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)
```

Methods

```
__init__(saxs_data, composition_data, ...)

change_model_callback(data)

change_norder_callback(figure, location, click)

draw_peaks(peaks)

goto_callback(click)

label(label)

next_button_callback(click)

run()

ternary_click_callback(figure, location, click)

update_plot()

__init__(saxs_data, composition_data, possible_phase_labels)

next_button_callback(click)
```

```
__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

 ${\bf class} \ \, {\tt AFL.automation.shared.DiffractionLabeler.DiffractionLabelerModel} (saxs_data, \\ composition_data, \\ possible_phase_labels)$

```
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()
```

```
ordinal_phase_labels()
init_models()
```

get_peaks(model, qstar=None, max_order=4)

AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView

class AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView
 Bases: object

__init__()

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 $un-known_value$ 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.

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.
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.
                 self – Estimator instance.
              Return type
                  estimator instance
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

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__()

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

```
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__()

Methods

init()	
clear()	
<pre>get(k[,d])</pre>	
items()	
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.

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.

 $extbf{popitem}() o (k, v)$, remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if \boldsymbol{D} is empty.

 ${\tt setdefault}(k\big[,d\,\big]) \to {\rm D.get(k,d)}, \text{ also set D[k]=d if k not in D}$

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

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')

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** (*boo1*) 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
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 date-time 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,])	
<pre>Zeroconf([interfaces, unicast, ip_version,])</pre>	Implementation of Zeroconf Multicast DNS Service Discovery

AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser

 $\textbf{class} \ \, \textbf{AFL.automation.shared.ServerDiscovery.} \\ \textbf{AsyncServiceBrowser}(\textit{zeroconf}; \ \textit{Zeroconf}, \textit{type_: str} \ | \ \textit{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 | 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
- 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.
<pre>async_update_records_complete()</pre>	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 | 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
- 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)	List addresses matching IP version.
async_clear_cache()	Clear the cache for this service info.
<pre>async_request(zc, timeout[, question_type,])</pre>	Returns true if the service could be discovered on the 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 discovered.
<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.
name
                                                 The name of the service.
                                                 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.shared.ServerDiscovery.AsyncZeroconf

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) \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 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)

Methods

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 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)

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.

```
{\tt async\_remove\_all\_service\_listeners()} \rightarrow {\tt 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.

Methods

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, 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) \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)
```

 $Return\ True\ if\ member\ is\ a\ member\ of\ this\ enum\ raises\ Type Error\ if\ member\ is\ not\ an\ enum\ member$

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

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

```
Bases: Thread
__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>	
<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.
on_service_state_change(zeroconf,)	
sa_aio_discover_server_by_name(service_name	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 \mid list, handlers: ServiceListener \mid list[Callable[..., None]] \mid None = None, listener: ServiceListener \mid None = None, addr: <math>str \mid None = None, port: int = 5353, delay: int = 10000, question\_type: DNSQuestionType \mid 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
- *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(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

__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
- 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.
	Returns true if the service could be discovered on the
<pre>async_request(zc, timeout[, question_type,])</pre>	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 discovered.
<pre>set_server_if_missing()</pre>	Set the server if it is missing.
update_record(zc, now, record)	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) → 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() \rightarrow None
```

Set the server if it is missing.

This function is for backwards compatibility.

```
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.ServiceStateChange

```
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]] | 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.
<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.

continues on next page

Table 60 – continued from previous page

	a nem previous page
async_update_service(info)	Registers service information to the network with a default TTL.
async_wait(timeout)	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>	
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.
<pre>send(out[, addr, port, v6_flow_scope, transport])</pre>	Sends an outgoing packet threadsafe.
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[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)

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 DNSQuestion-Type.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) → 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 <code>async_update_service</code> 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 \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)

```
\begin{tabular}{ll} \begin{tabular}{ll} \textbf{generate\_service\_broadcast}(info: ServiceInfo, ttl: int \mid None, broadcast\_addresses: bool = True) \rightarrow \\ DNSOutgoing \end{tabular}
```

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) \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.

```
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 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 \\ \textbf{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) <math>\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) <math>\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.

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)

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

 $\textbf{exception} \hspace{0.1cm} \textbf{AFL.automation.shared.exceptions.} \textbf{NoDeviceFoundException} \\$

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

AFL.automation.shared.exceptions.NotFoundError

exception AFL.automation.shared.exceptions.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

enforce_units(value, unit_type)	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

AFL.automation.shared.utilities.tprint(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

clear_output([wait])	Clear the output of the current cell receiving output.
<pre>client_construct_ui(self[, return_ui,])</pre>	
<pre>display(*objs[, include, exclude, metadata,])</pre>	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,
    "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.

 ${\tt AFL.automation.shared.widgetui.client_construct_ui} (\textit{self}, \textit{return_ui} = \textit{False}, \textit{display_ui} = \textit{True})$

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()

AFL.automation.test()

Run all tests using pytest.

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

```
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 791

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')
```

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

Communicate with NistoRoboto server on OT-2

```
__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 ().

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.

7.1. AFL.automation 793

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.

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.

7.1. AFL.automation 795

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.

__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

```
а
                                                                                                               41
                                                                                               AFL.automation.loading.PressureController, 42
AFL.automation, 13
                                                                                               AFL.automation.loading.PressureControllerAsPump,
AFL.automation.APIServer, 14
AFL.automation.APIServer.APIServer, 14
                                                                                               AFL.automation.loading.PushPullSelectorSampleCell,
AFL.automation.APIServer.Client, 17
AFL.automation.APIServer.Driver, 19
                                                                                               AFL.automation.loading.RSoXSSolutionSampleCell,
AFL.automation.APIServer.DummyDriver, 20
AFL.automation.APIServer.DummyOT2Driver, 21
                                                                                               AFL.automation.loading.SainSmartRelay, 47
AFL.automation.APIServer.LoggerFilter, 22
                                                                                               AFL.automation.loading.SampleCell, 48
AFL.automation.APIServer.QueueDaemon, 22
                                                                                               AFL.automation.loading.Sensor, 48
AFL.automation.EpicsADLiveProcess, 23
\textbf{AFL.} automation. Epics \textbf{ADL} ive \textbf{Process.AreaDetector} \\ \textbf{Elve} \\ \textbf{automation.loading.Sensor} \\ \textbf{CallbackThread}, \\ \textbf{Callba
                                                                                               AFL.automation.loading.SensorPollingThread,
AFL.automation.EpicsADLiveProcess.Client, 24
AFL.automation.EpicsADLiveProcess.ReduceDaemon,
                                                                                               AFL.automation.loading.SerialDevice, 53
                                                                                               AFL.automation.loading.SyringePump, 53
AFL.automation.instrument, 25
                                                                                               AFL.automation.loading.Tubing, 54
AFL.automation.instrument.DummySAS, 25
                                                                                               AFL.automation.loading.TwoSelectorBlowoutSampleCell,
AFL.automation.instrument.FileCamera, 26
AFL.automation.instrument.I22SAXS, 26
                                                                                               AFL.automation.loading.UltimusVPressureController,
AFL.automation.instrument.NetworkCamera, 27
AFL.automation.instrument.SeabreezeUVVis, 27
                                                                                               AFL.automation.loading.ViciMultiposSelector,
AFL.automation.instrument.SpecScreen_Driver,
                                                                                               AFL.automation.prepare, 57
AFL.automation.loading, 29
                                                                                               AFL.automation.prepare.Component, 58
AFL.automation.loading.CetoniMultiPosValve,
                                                                                               AFL.automation.prepare.DeckBuilderWidget, 59
                                                                                               AFL.automation.prepare.Dummy_OT2_Driver, 61
AFL.automation.loading.ChemyxSyringePump, 31
AFL.automation.loading.DigitalOutPressureControlleautomation.prepare.factory,68
                                                                                               AFL.automation.prepare.OT2Client, 62
{\tt AFL. automation. loading. Double ViciMultipos Selector}; automation. {\tt prepare. Prepare Widget}, 63
                                                                                               AFL.automation.prepare.PrepType, 63
                                                                                               AFL.automation.prepare.SampleSeriesWidget, 64
AFL.automation.loading.DummyPump, 35
                                                                                               AFL.automation.prepare.StockBuilderWidget,66
AFL.automation.loading.FlowSelector, 35
                                                                                               AFL.automation.prepare.SweepBuilderWidget, 67
AFL.automation.loading.LoadStopperDriver, 35
                                                                                               AFL.automation.prepare.utilities, 69
AFL.automation.loading.MultiChannelRelay, 36
                                                                                               AFL.automation.sample, 82
AFL.automation.loading.NE1kSyringePump, 37
{\tt AFL.automation.loading.OneSelectorBlowoutSample Celay to mation.sample.Casting Server, 82} \\
                                                                                               AFL.automation.sample_env, 69
{\tt AFL. automation. loading. Pneumatic Pressure Sample} \underbrace{{\tt AFL. automation. sample\_env. Temperature Deck}}_{interpretation}. \\
                                                                                               AFL.automation.shared, 70
                                                                                               AFL.automation.shared.DataLabelerWidget,70
AFL.automation.loading.PneumaticSampleCell,
```

```
AFL.automation.shared.DatasetWidget, 72
AFL.automation.shared.DiffractionLabeler, 74
AFL.automation.shared.exceptions, 79
AFL.automation.shared.MutableQueue, 75
AFL.automation.shared.PersistentConfig, 76
AFL.automation.shared.serialization, 80
AFL.automation.shared.ServerDiscovery, 77
AFL.automation.shared.units, 80
AFL.automation.shared.utilities, 81
AFL.automation.shared.widgetui, 81
```

800 Python Module Index

INDEX

Symbols	method), 508
bytes() (AFL.automation.instrument.SeabreezeUVVis.Pdffl	() (AFL.automation.prepare.PrepareWidget.Layout
method), 202	method), 519
call() (AFL.automation.APIServer.APIServer.Flaskdel	() (AFL.automation.prepare.PrepareWidget.Text
method) 127	<i>method</i>), 531
contains() (AFL.automation.APIServer.APIServer.IPvdelon	() (AFL.automation.prepare.PrepareWidget.VBox
class method) 127	memoa), 559
contains() (AFL.automation.prepare.Component.Prepf)pe_	() (AFL.automation.prepare.SampleSeriesWidget.Button
class method) 382	memoa), 548
contains() (AFL.automation.prepare.PrepType.Enum_del	() (AFL.automation.prepare.SampleSeriesWidget.Checkbox
class mathod) 472	<i>methoa</i>), 550
contains() (AFL.automation.prepare.PrepType.PrepType^l	() (AFL.automation.prepare.SampleSeriesWidget.FloatText
class mathod) 172	meinoa), 507
contains() (AFL.automation.shared.ServerDiscovery.H ^{fl} versia	(AFL.automation.prepare.SampleSeriesWidget.HBox
class method), 760	method), 574
contains() (AFL.automation.shared.ServerDiscovery.Services	
class method), 773	method), 582
del() (AFL.automation.prepare.DeckBuilderWidget.Button1	
	method), 590
method), 387del() (AFL.automation.prepare.DeckBuilderWidget.CheckBox	() (AFL.automation.prepare.SampleSeriesWidget,Layout
del() (AFL.automation.prepare.Deckbuttaerwitaget.Cneckbox	method), 600
method), 395del() (AFL.automation.prepare.DeckBuilderWidget.Dropadown	() (AFL automation prepare SampleSeriesWidget Text
del() (AFL.automation.prepare.DeckBuilderwiaget.Dropaown	method), 610
method), 408 del() (AFL.automation.prepare.DeckBuilderWidget.HBdel	() (AFL automation prepare SampleSeriesWidget VRox
del() (AFL.automation.prepare.DeckBuilderWiaget.HBox =	method), 618
method), 416 del() (AFL.automation.prepare.DeckBuilderWidget.Labdel	() (AFL automation prepare SweepRuilderWidget Rutton
del() (AFL.automation.prepare.DeckBuilderWiaget.L abet	method), 631
method), 423del() (AFL.automation.prepare.DeckBuilderWidget.Layout —	() (AFL automation prepare SweenRuilderWidget Checkbox
	method), 639
method), 434 del() (AFL.automation.prepare.DeckBuilderWidget.Text ^{del}	() (AFL automation prepare SweepRuilderWidget HRox
del() (AFL.automation.prepare.DeckBuilderWidget.Text	method), 646
method), 441	() (AFI automation prepare SweenRuilderWidget Lahel
del() (AFL.automation.prepare.DeckBuilderWidget.VBodel	method), 654
method), 449	() (AFI automation prepare SweenRuilderWidget Layout
del() (AFL.automation.prepare.PrepareWidget.Buttondel	method), 664
method), 477	
del() (AFL.automation.prepare.PrepareWidget.Checkboxdel	method), 674
method), 484	() (AFL automation prepare Sween Puilder Widget VPor
del() (AFL.automation.prepare.PrepareWidget.Dropdown del	mathod) 691
method) 493	memoa), 001
del() (AFL.automation.prepare.PrepareWidget.HBoxeq() (AFL:automation.prepare.Solution method), 377
method), 501	3//
del() (AFL.automation.prepare.PrepareWidget.Label—_eq() (AFL.automation.prepare.factory.Solution

method), 690	method), 154
getitem() (AFL.automation.APIServer.APIServer.IPVerinint_	_() (AFL.automation.APIServer.Driver.Driver
class method), 128	method), 19, 157, 158, 162
getitem() (AFL.automation.APIServer.Driver.Persistentistaints	g_() (AFL.automation.APIServer.Driver.PersistentConfig
method), 161	method), 160, 161
getitem() (AFL.automation.prepare.Component.PrepTypmit_	() (AFL.automation.APIServer.DummyDriver.Driver
class method), 383	method), 164
getitem() (AFL.automation.prepare.PrepType.Enuminit_	_() (AFL.automation.APIServer.DummyDriver.DummyDriver
class method), 472	method), 21, 166, 168, 169
getitem() (AFL.automation.prepare.PrepType.PrepTypainit_	_() (AFL.automation.APIServer.DummyOT2Driver.Driver
class method), 472	method), 170, 171
getitem() (AFL.automation.shared.PersistentConfig.Perisisten	tConfigFL.automation.APIServer.DummyOT2Driver.DummyDriver
method), 77, 746, 748	method), 21, 172, 174, 175
getitem() (AFL.automation.shared.ServerDiscovery.IP\\arista	n_() (AFL.automation.APIServer.LoggerFilter.LoggerFilter
class method), 760	method), 22, 176
getitem() (AFL.automation.shared.ServerDiscovery.SerimatS	taleXlAdrigautomation.APIServer.QueueDaemon.DataTrashcan
class method), 773	method), 177, 178
hash() (AFL.automation.prepare.Component.Componeninit_	_() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
method), 59, 381, 383	method), 23, 179, 180, 182
hash() (AFL.automation.prepare.Solute method),init_	$_$ () (AFL.automation. Epics ADLive Process. Area Detector Live. Area
375	method), 23, 790, 791
hash() (AFL.automation.prepare.Solutioninit_	_() (AFL.automation.EpicsADLiveProcess.Client.Client
method), 377	method), 24, 791, 792
hash() (AFL.automation.prepare.Solvent method),init_	$_()$ (AFL.automation. Epics ADL ive Process. Reduce Daemon. Redu
379	method), 25, 793, 794, 796
hash() (AFL.automation.prepare.factory.Solutioninit_	_() (AFL.automation.instrument.DummySAS.Driver
method), 690	method), 183, 184
init() (AFL.automation.APIServer.APIServer.APIServeinit_	_() (AFL.automation.instrument.DummySAS.DummySAS
method), 16, 90, 91, 149	method), 26, 185, 186, 188
init() (AFL.automation.APIServer.APIServer.CORSinit_	_() (AFL.automation.instrument.FileCamera.FileCamera
method), 95	method), 26, 188
init() (AFL.automation.APIServer.APIServer.FileHandlanit_	
method), 95, 96	method), 189
init() (AFL.automation.APIServer.APIServer.Flaskinit_	
method), 99, 105	method), 26, 191–193
init() (AFL.automation.APIServer.APIServer.IPVersioninit_	
method), 127	method), 27, 194
init() (AFL.automation.APIServer.APIServer.JWTManaigeirt_	
method), 128, 129	method), 194, 195
init() (AFL.automation.APIServer.APIServer.Logger <u>Fil</u> terit_	
method), 132, 133	method), 197
init() (AFL.automation.APIServer.APIServer.MutableQurine_	
method), 133	method), 197
init() (AFL.automation.APIServer.APIServer.QueueDaimioto_	
method), 134	method), 27, 203, 205, 207
init() (AFL.automation.APIServer.APIServer.SMTPHandlet_	· · · · · · · · · · · · · · · · · · ·
method), 136, 137	method), 208, 209
init() (AFL.automation.APIServer.APIServer.ServiceInfinit_	· · · · · · · · · · · · · · · · · · ·
method), 139, 141	method), 28, 210–212
init() (AFL.automation.APIServer.APIServer.Zeroconfinit_	· · · · · · · · · · · · · · · · · · ·
method), 144, 146	method), 31, 215, 216
init() (AFL.automation.APIServer.Client.Clientinit_	
method), 17, 151, 152, 155	method), 216

__init__() (AFL.automation.APIServer.Client.ServerDiscovinyit__() (AFL.automation.loading.ChemyxSyringePump.ChemyxConn

```
method), 33, 217, 218, 222
                                                                method), 39, 273, 275, 279
__init__() (AFL.automation.loading.ChemyxSyringePump.GlnertyxSy) in Penantomation.loading.PneumaticPressureSampleCell.Sam
        method), 32, 219–221
                                                                method), 277
__init__() (AFL.automation.loading.ChemyxSyringePump.SymintgePumpAFL.automation.loading.PneumaticPressureSampleCell.defa
        method), 221
                                                                method), 277, 278
__init__() (AFL.automation.loading.DigitalOutPressureController.D)giAHDuutPressuiveCloadiohler.PneumaticSampleCell.Driver
        method), 34, 223–225
                                                                method), 281
__init__() (AFL.automation.loading.DigitalOutPressureCoritmalter.R)@(Adrik.Gattmalteon.loading.PneumaticSampleCell.PneumaticSa
        method), 224
                                                                method), 41, 283, 285, 288
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Selivnixt,DA</u>)b(AVIL:MadbipasSalkaading.PneumaticSampleCell.SampleCell
        method), 34, 226, 228
                                                                method), 286
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Seliviotr.Fl6</u>)vS&leEtantomation.loading.PneumaticSampleCell.defaultdict
        method), 227
                                                                method), 287, 288
__init__() (AFL.automation.loading.DoubleViciMultipos<u>Selinivt.Vi¢iMAlfipanSoluctioon.</u>loading.PressureController.PressureControl
        method), 227, 228
                                                                method), 290
__init__() (AFL.automation.loading.DummyPump.DummyPump_() (AFL.automation.loading.PressureControllerAsPump.Pressur
        method), 35, 229, 231
                                                                method), 43, 291–293
__init__() (AFL.automation.loading.DummyPump.SerialDevicet__() (AFL.automation.loading.PressureControllerAsPump.SerialD
                                                                method), 292, 293
        method), 230
__init__() (AFL.automation.loading.DummyPump.SyringePimixt__() (AFL.automation.loading.PressureControllerAsPump.Syringe
        method), 230
                                                                method), 293
__init__() (AFL.automation.loading.FlowSelector.FlowSelectarit__() (AFL.automation.loading.PushPullSelectorSampleCell.Driver
                                                                method), 294, 295
        method), 232
__init__() (AFL.automation.loading.LoadStopperDriver.Clizmit__() (AFL.automation.loading.PushPullSelectorSampleCell.PushF
                                                                method), 44, 297, 299, 304
        method), 233, 234
__init__() (AFL.automation.loading.LoadStopperDriver.Drinit__() (AFL.automation.loading.PushPullSelectorSampleCell.Sampl
        method), 235, 236
                                                                method), 301
__init__() (AFL.automation.loading.LoadStopperDriver.LoadStopp@DtAFdr.automation.loading.PushPullSelectorSampleCell.Tubing
        method), 36, 238, 240, 250
                                                                method), 301, 302
__init__() (AFL.automation.loading.LoadStopperDriver.<u>SerismiRoll</u>ing(AhFdadutomation.loading.PushPullSelectorSampleCell.defaul
        method), 241, 242
                                                                method), 302, 303
__init__() (AFL.automation.loading.LoadStopperDriver.StophindCDylAFL.automation.loading.RSoXSSolutionSampleCell.Driver
        method), 244, 245
                                                                method), 305, 306
__init__() (AFL.automation.loading.LoadStopperDriver.<u>StophirtadCB</u>yQAFL.automation.loading.RSoXSSolutionSampleCell.RSoXSS
        method), 247, 248
                                                                method), 46, 308, 310, 315
__init__() (AFL.automation.loading.MultiChannelRelay.MultiChann@lRelayautomation.loading.RSoXSSolutionSampleCell.SampleC
        method), 251
                                                                method), 312
__init__() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePuAlfL.automation.loading.RSoXSSolutionSampleCell.Tubing
        method), 37, 252, 253, 255
                                                                method), 312, 313
__init__() (AFL.automation.loading.NE1kSyringePump.SeriadDevice) (AFL.automation.loading.RSoXSSolutionSampleCell.defaultd
        method), 254
                                                                method), 313, 314
__init__() (AFL.automation.loading.NE1kSyringePump.SyringePum(i) (AFL.automation.loading.SainSmartRelay.MultiChannelRelay
        method), 254
                                                                method), 317
__init__() (AFL.automation.loading.OneSelectorBlowoutSainpiltCeNDtAFdr.automation.loading.SainSmartRelay.SainSmartRelay
                                                                method), 47, 318, 319
        method), 256, 257
__init__() (AFL.automation.loading.OneSelectorBlowout<u>SairpiltCeNQAASelautonBltivroutoSalinpl</u>eGaethSmartRelay.SerialDevice
        method), 38, 259, 260, 269
                                                                method), 319
__init__() (AFL.automation.loading.OneSelectorBlowout<u>Sa</u>inpilt<u>CeNJVAASelautonRitivonUtSainpileSainpileCell</u>.SampleCell
        method), 263, 265
                                                                method), 320
__init__() (AFL.automation.loading.OneSelectorBlowout<u>SainpiltCellyleAtMldiat</u>omation.loading.Sensor.DummySensor1
        method), 267, 268
                                                                method), 49, 321, 322, 327
__init__() (AFL.automation.loading.PneumaticPressureSamipticCell(DriAFL.automation.loading.Sensor.DummySensor2
        method), 270, 271
                                                                method), 49, 324, 325, 328
__init__() (AFL.automation.loading.PneumaticPressureSaniplicCell(BneumAfidPausomasampledinlg.Sensor.Sensor
```

	method), 48, 327	method), 60, 404, 455
init_	_() (AFL.automation.loading.SensorCallbackThreadilmiate	e rConAnFilnination ation.prepare.DeckBuilderWidget.DeckBuilderWi
	method), 52, 328, 329, 343	method), 60, 404, 405, 455
init		or Cdl(Addk:Thutomi ttion.prepare.DeckBuilderWidget.DeckBuilderWi
	method), 50, 329, 330, 341	method), 60, 405, 455
ini+		Le (I) r(AlFdldGB)mation.prepare.DeckBuilderWidget.Dropdown
	method), 52, 332, 333, 342	method), 406, 408
1n1t_	() (AFL.automation.loading.SensorCallbackThre <u>adiStopL</u>	
	method), 51, 335, 336, 341	method), 414, 416
init_	_() (AFL.automation.loading.SensorCallbackThre <u>adistopL</u>	
	method), 51, 338, 339, 342	method), 422, 424
init_	() (AFL.automation.loading.SensorPollingThread.Sinstr	PollingTHr.audomation.prepare.DeckBuilderWidget.Layout
	method), 52, 343, 344, 346	method), 429, 434
init	_() (AFL.automation.loading.SerialDevice.Serial <u>Dev</u> ina t_	
	method), 53, 347	method), 439, 441
ini+	() (AFL.automation.loading.SyringePump.SyringeP imi rt_	
	method), 348	method), 448, 449
init_		() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
	method), 54, 349	method), 456
init_	() (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpilat <u>(</u>	<u>Ce</u> N.D.A.Felr.automation.prepare.Dummy_OT2_Driver.Dummy_OT2
	method), 350, 351	method), 61, 458, 459, 461
init_	() (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpilt(Cell Sampl ACU lautomation.prepare.MassBalance
	method), 352	method), 370, 371
init	() (AFL.automation.loading.TwoSelectorBlowout <u>Sa</u> inpile(
	method), 353	method), 463, 464
ini+		
	() (AFL.automation.loading.TwoSelectorBlowout <u>SainpileC</u>	
	method), 55, 354, 356, 360	method), 466, 468
1n1t_	_() (AFL.automation.loading.TwoSelectorBlowout <u>Sainpile(</u>	
	method), 358, 359	method), 469
init_	_() (AFL.automation.loading.UltimusVPressureCont ivol ler_	Pte rsure ConExalleo mation.prepare.PipetteAction
	method), 362	method), 372
init_	() (AFL.automation.loading.UltimusVPressureCont ivo ll&r_	<u>U(t)</u> mu(A/H lr austone/Gtiotr.phlep are.PrepType.Enum
	method), 56, 362–364	method), 472
init	_() (AFL.automation.loading.ViciMultiposSelector.Fiorise	
	method), 364	method), 472
ini+	() (AFL.automation.loading.ViciMultiposSelector <u>.Sa</u> niahD	
	method), 365	method), 473
1n1t_	_() (AFL.automation.loading.ViciMultiposSelecto <u>r.ViciMu</u>	
	method), 57, 365, 366	method), 474, 476
init_	() (AFL.automation.prepare.Component.Compon <u>en</u> init_	
	method), 59, 381, 383	method), 482, 484
init_	() (AFL.automation.prepare.Component.PrepType_init_	() (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget
	method), 382	method), 490
init		_() (AFL.automation.prepare.PrepareWidget.Dropdown
	method), 368	method), 491, 493
ini+		_() (AFL.automation.prepare.PrepareWidget.HBox
	* * *	
	369	method), 499, 501
init_	() (AFL.automation.prepare.DeckBuilderWidget. <u>Buitmi</u> t_	
	method), 385, 387	method), 507, 509
init_	() (AFL.automation.prepare.DeckBuilderWidget. <u>Ch</u> irni bto	x_() (AFL.automation.prepare.PrepareWidget.Layout
	method), 393, 395	method), 514, 519
init_		_() (AFL.automation.prepare.PrepareWidget.PrepareWidget
_	method), 401, 402	method), 64, 524, 545
init		ld&)WAGetautomation.prepare.PrepareWidget.PrepareWidget_Mo
	() (III Landonanon.propare.DeckDanaer waget.Detailed)	wan and in the state of the sta

	method), 524	method), 67, 627, 628
init_	() (AFL.automation.prepare.PrepareWidget.Prep <u>are</u> Widg	et (VieAFL.automation.prepare.SweepBuilderWidget.Button
	method), 525	method), 629, 631
init	**	W(dg&FL.automation.prepare.SweepBuilderWidget.Checkbox
	method), 525, 526	method), 637, 639
init	() (AFL.automation.prepare.PrepareWidget.StockBrithdet)	
	method), 527	method), 645, 646
init	() (AFL.automation.prepare.PrepareWidget.Sweep Bunkh e	
	method), 528	method), 652, 654
init	() (AFL.automation.prepare.PrepareWidget.Textinit_	
	method), 529, 531	method), 660, 664
init		() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder
	method), 538, 539	method), 67, 669, 670, 687
init		() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder
	373	method), 68, 670, 687
init_		() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder
	method), 373, 374	method), 68, 671, 687
ini+	() (AFL.automation.prepare.SampleSeriesWidget <u>.Bimir</u> t_	
	method), 546, 548	method), 671, 673
ini+	() (AFL.automation.prepare.SampleSeriesWidget <u>.Chrekb</u>	
1111	method), 554, 556	method), 680, 681
ini+	() (AFL.automation.prepare.SampleSeriesWidget <u>.Clivit</u> t_	
1111 (_	method), 562, 563	method), 689, 690
ini+	() (AFL.automation.prepare.SampleSeriesWidget <u>.FlivaiTe</u> .	
1111 (_	() (AT L. automation. prepare. Sample Series wiaget <u>. From he</u> method), 565, 567	method), 691
ini+	() (AFL.automation.prepare.SampleSeriesWidget .HBni t_	
1111	method), 573, 575	method), 82, 693, 695, 705
init	() (AFL.automation.prepare.SampleSeriesWidget <u>.IniFe</u> ix t _	
1111	method), 580, 582	method), 696, 698
init	() (AFL.automation.prepare.SampleSeriesWidget <u>.Lairei</u> t_	
	method), 588, 590	method), 699, 700
init	() (AFL.automation.prepare.SampleSeriesWidget <u>.Lai</u> xaiut_	
	method), 596, 600	method), 701, 704
init	() (AFL.automation.prepare.SampleSeriesWidget <u>.Saimple</u>	
	method), 65, 605, 606, 624	method), 706
init		<u>Sefix (MFdgatut)Modti</u> bn.sample_env.TemperatureDeck.TemperatureL
	method), 66, 607, 624	method), 69, 708–710
init		<u>Se(i)e(WildgatutVinu</u> tion.shared.DataLabelerWidget.DataLabelerMo
	method), 66, 607, 608, 624	method), 71, 712, 722
init_		() (AFL.automation.shared.DataLabelerWidget.DataLabelerVie
	method), 608, 610	method), 71, 712, 722
init		() (AFL.automation.shared.DataLabelerWidget.DataLabelerWid
	method), 617, 618	method), 70, 713, 714, 721
init_		() (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
	374, 375	method), 718
init_		() (AFL.automation.shared.DatasetWidget.DatasetWidget
	method), 376, 377	method), 72, 722, 724, 730
init_		() (AFL.automation.shared.DatasetWidget.DatasetWidget_Mode
	378, 379	method), 73, 726, 731
init_		ilde WAd Jetautomation.shared.Dataset Widget.Dataset Widget_View
	method), 66, 625–627	method), 74, 727, 732
init_	() (AFL.automation.prepare.StockBuilderWidget. <u>StoickBui</u>	
	method), 67, 626–628	method), 728, 729

 $\verb|__init__()| (AFL. automation. prepare. Stock Builder Widget. \underline{Stock Build(2)} \ WAF \underline{Jexta_Wirm} \\ at ion. shared. Diffraction Labeler. Diffraction La$

	method), 74, 733, 741	class method), 773
init_	_() (AFL.automation.shared.DiffractionLabeler.Diffr heti on	LdhelFiMndemation.APIServer.APIServer.IPVersion
	method), 75, 734, 741	class method), 128
init_	_() (AFL.automation.shared.DiffractionLabeler.Diffr hen on	Label Fil Viest omation, prepare, Component, PrepType
	method), 75, 734, 741	class method), 383
init	_() (AFL.automation.shared.DiffractionLabeler.Or <u>dihetEnc</u>	
	method), 738	class method), 472
init	_() (AFL.automation.shared.MutableQueue.Mutable Qua ue	
	method), 76, 742, 743	class method), 473
init	_() (AFL.automation.shared.PersistentConfig.Mutable 41 app	
	method), 744	class method), 760
ini+		fig(AFL.automation.shared.ServerDiscovery.ServiceStateChange
1111	method), 76, 745–747	class method), 773
init		
1II1 t_		<u>Panow</u> (∂r(AFL.automation.APIServer.Driver.PersistentConfig
	method), 749, 750	method), 161
1n1t_		<u>emfo_()</u> (AFL.automation.shared.PersistentConfig.PersistentConfig
	method), 751, 753	method), 77, 746, 748
init_	_() (AFL.automation.shared.ServerDiscovery.Asyn <u>cZetvco</u>	
	method), 756, 757	method), 202
init_	_() (AFL.automation.shared.ServerDiscovery.AsyncZerocom	nfServiceTypes
	method), 758, 759	
init_	_() (AFL.automation.shared.ServerDiscovery.IPVersionter	(AFL.automation.APIServer.APIServer.Flask
	method), 760	attribute), 106
init_	_() (AFL.automation.shared.ServerDiscovery.Run Thrace er	_class(AFL.automation.APIServer.APIServer.Flask
	method), 78, 760, 761, 779	attribute), 103
init_	_() (AFL.automation.shared.ServerDiscovery.ServerDiscovery	(BC) (AFL, automation instrument Seabreeze UVV is Path
	method), 78, 763, 779	method), 200
init_	_() (AFL.automation.shared.ServerDiscovery.ServiceBrows	(AFL automation APIServer APIServer FileHandler
	method), 764, 766	method), 96
init_	_() (AFL.automation.shared.ServerDiscovery.ServiceInfre	() (AFL automation APIServer APIServer SMTPHandler
	method), 769, 770	method), 137
init	_() (AFL.automation.shared.ServerDiscovery.ServiceStateA	MUNRO mation prepare Component DR method)
	method), 773	368
init		ay() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
	method), 774, 775	ay()(AFL.automation.AFIServer.QueueDaemon.DataTrasncan
init	_() (AFL.automation.shared.widgetui.Markdown add_cat	method), 178
	method), 787, 788	
itor		370
1.61_	_() (AFL.automation.APIServer.APIServer.IPVersignd_cla	
:+~~	class method), 128	method), 388
rer_	_() (AFL.automation.prepare.Component.Compon eadd_ Cla	ss() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
	method), 59, 381, 384	method), 395
iter_	_() (AFL.automation.prepare.Component.Prep1ypadd_cla	ss() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
	class method), 383	method), 408
iter_	_() (AFL.automation.prepare.PrepType.Enum add_cla	ss() (AFL.automation.prepare.DeckBuilderWidget.HBox
_	class method), 472	method), 416
iter_	_() (AFL.automation.prepare.PrepType.PrepType add_cla	ss() (AFL.automation.prepare.DeckBuilderWidget.Label
	class method), 473	method), 424
iter_	_() (AFL.automation.prepare.Solute method), add_cla	ss() (AFL.automation.prepare.DeckBuilderWidget.Text
	375	method), 441
iter_	_() (AFL.automation.prepare.Solvent method), add_cla	ss() (AFL.automation.prepare.DeckBuilderWidget.VBox
	379	method) 449
iter_	_() (AFL.automation.shared.ServerDiscovery.IPVerdip_ncla	ss() (AFL.automation.prepare.PrepareWidget.Button
	class method), 760	method), 477
iter	_() (AFL.automation.shared.ServerDiscovery.ServiceStateC	

```
add_class() (AFL.automation.prepare.PrepareWidget.Chaddopipette()
                                                                                                                     (AFL.automation.prepare.Deck
                                                                                               method), 370
             method), 484
add_class() (AFL.automation.prepare.PrepareWidget.Draphbyprep_targets() (AFL.automation.prepare.Dummy OT2 Driver.Du.
                                                                                               method), 61, 459, 462
             method), 493
add_class() (AFL.automation.prepare.PrepareWidget.HBadd_sample() (AFL.automation.prepare.SampleSeries
             method), 501
                                                                                               method), 374
add_class() (AFL.automation.prepare.PrepareWidget.Labadd_service() (AFL.automation.shared.ServerDiscovery.AsyncZeroconfl
             method), 509
                                                                                               method), 759
add_class() (AFL.automation.prepare.PrepareWidget.Textadd_service_listener()
                                                                                               (AFL.automation.APIServer.APIServer.Zeroconf
             method), 531
add_class() (AFL.automation.prepare.PrepareWidget.VBox
                                                                                               method), 147
                                                                                 add_service_listener()
             method), 539
add_class() (AFL.automation.prepare.SampleSeriesWidget.Button (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                               method), 776
             method), 549
add_class() (AFL.automation.prepare.SampleSeriesWidgetddhectdandard_routes()
             method), 556
                                                                                               (AFL.automation.APIServer.APIServer.APIServer
add_class() (AFL.automation.prepare.SampleSeriesWidget.FloatTextethod), 16, 92, 149
             method), 567
                                                                                 add_stock() (AFL.automation.prepare.Deck method),
add_class() (AFL.automation.prepare.SampleSeriesWidget.HBox 370
             method), 575
                                                                                 add_stock()
                                                                                                         (AFL.automation.prepare.MassBalance
add_class() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 371
             method), 582
                                                                                 add_stocks_to_deck()
add_class() (AFL.automation.prepare.SampleSeriesWidget.Label (AFL.automation.prepare.PrepareWidget.StockBuilderWidget
                                                                                               method), 528
             method), 590
add_class()(AFL.automation.prepare.SampleSeriesWidg@rdLexstocks_to_deck()
             method), 610
                                                                                               (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidge
add_class() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 66, 626, 627
                                                                                 add_stocks_to_deck()
             method), 618
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), 67, 627, 628
             method), 631
add_class() (AFL.automation.prepare.SweepBuilderWidgatdChreklget() (AFL.automation.prepare.Deck method),
             method), 639
                                                                                               370
add_class() (AFL.automation.prepare.SweepBuilderWidgadblBcemplate_filter()
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 646
add_class() (AFL.automation.prepare.SweepBuilderWidget.Label method), 113
                                                                                 add_template_global()
             method), 654
add_class() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 674
                                                                                               method), 114
add_class()(AFL.automation.prepare.SweepBuilderWidgatd\Bommplate_test()
             method), 681
                                                                                               (AFL.automation.APIServer.APIServer.Flask
add_component_from_name()
                                                                                               method), 113
             (AFL.automation.prepare.factory.Solution
                                                                                 add_traits() (AFL.automation.prepare.DeckBuilderWidget.Button
             method), 690
                                                                                               method), 388
add_component_from_name()
                                                                                 add_traits() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
             (AFL.automation.prepare.Solution
                                                                  method),
                                                                                               method), 395
             377
                                                                                 add_traits() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
add_container()
                                   (AFL.automation.prepare.Deck
                                                                                               method), 408
                                                                                 add_traits()(AFL.automation.prepare.DeckBuilderWidget.HBox
             method), 370
add_interactive() (AFL.automation.prepare.ComponentDB
                                                                                               method), 416
             method), 368
                                                                                 add\_traits() (AFL.automation.prepare.DeckBuilderWidget.Label
add_listener()(AFL.automation.APIServer.APIServer.Zeroconf method), 424
                                                                                 add_traits()(AFL.automation.prepare.DeckBuilderWidget.Layout
             method), 148
add_listener() (AFL.automation.shared.ServerDiscovery.Zeroconfinethod), 434
             method), 778
                                                                                 add_traits() (AFL.automation.prepare.DeckBuilderWidget.Text
```

```
method), 442
                                                                                                              add_url_rule() (AFL.automation.APIServer.APIServer.Flask
add_traits()(AFL.automation.prepare.DeckBuilderWidget.VBox method), 111
                  method), 449
                                                                                                               add_vertical_line()
                                                                                                                                 (AFL.automation.shared.DataLabelerWidget.DataLabelerView
add_traits() (AFL.automation.prepare.PrepareWidget.Button
                  method), 477
                                                                                                                                 method), 71, 713, 722
add_traits()(AFL.automation.prepare.PrepareWidget.Chaddbwertical_line()
                                                                                                                                 (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerVi
                  method), 484
add_traits()(AFL.automation.prepare.PrepareWidget.Dropdown method), 75, 735, 742
                  method), 493
                                                                                                               Added (AFL.automation.shared.ServerDiscovery.ServiceStateChange
add\_traits() (AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                 attribute), 773
                  method), 501
                                                                                                               addFilter() (AFL. automation. APIServer. APIServer. FileHandler
add_traits() (AFL.automation.prepare.PrepareWidget.Label
                                                                                                                                 method), 96
                  method), 509
                                                                                                               addFilter() (AFL.automation.APIServer.APIServer.SMTPHandler
add_traits() (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                 method), 137
                                                                                                               additional_claims_loader()
                  method), 519
add_traits() (AFL.automation.prepare.PrepareWidget.Text
                                                                                                                                 (AFL.automation.APIServer.APIServer.JWTManager
                  method), 532
                                                                                                                                 method), 130
add_traits()(AFL.automation.prepare.PrepareWidget.VMadditional_headers_loader()
                                                                                                                                 (AFL.automation.APIServer.APIServer.JWTManager
                  method), 539
add_traits() (AFL.automation.prepare.SampleSeriesWidget.Buttonmethod), 130
                  method), 549
                                                                                                              addMode() (AFL.automation.loading.ChemyxSyringePump.ChemyxConnec
add_traits() (AFL.automation.prepare.SampleSeriesWidget.Checkbroathod), 33, 219, 223
                                                                                                               addresses (AFL. automation. API Server. API Server. Service Info
                  method), 556
add_traits() (AFL.automation.prepare.SampleSeriesWidget.FloatTaxtribute), 141
                                                                                                              addresses (AFL. automation. shared. Server Discovery. Async Service Info
                  method), 567
add_traits() (AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 753
                  method), 575
                                                                                                               addresses (AFL. automation. shared. Server Discovery. Service Info
add_traits() (AFL.automation.prepare.SampleSeriesWidget.IntTextattribute), 770
                                                                                                               addresses_by_version()
                  method), 582
\verb|add_traits()| (AFL. automation. prepare. Sample Series Widget. Label (AFL. automation. API Server. API Server. Service Information. API Server. AP
                  method), 590
                                                                                                                                 method), 142
add_traits()(AFL.automation.prepare.SampleSeriesWidgeddhessses_by_version()
                                                                                                                                 (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                  method), 600
add_traits()(AFL.automation.prepare.SampleSeriesWidget.Text method), 754
                  method), 611
                                                                                                               addresses_by_version()
add_traits() (AFL.automation.prepare.SampleSeriesWidget.VBox (AFL.automation.shared.ServerDiscovery.ServiceInfo
                  method), 618
                                                                                                                                 method), 771
add_traits() (AFL.automation.prepare.SweepBuilderWidgetdE@y.6AFL.automation.loading.ChemyxSyringePump.ChemyxConnection
                  method), 631
                                                                                                                                 method), 33, 219, 223
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), 639
add_traits() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 66, 607, 624
                                                                                                               advanceSample() (AFL.automation.loading.PneumaticPressureSampleCe
                  method), 646
add_traits() (AFL.automation.prepare.SweepBuilderWidget.Label method), 40, 275, 280
                  method), 654
                                                                                                              advertise_zeroconf()
add_traits() (AFL.automation.prepare.SweepBuilderWidget.Layou(AFL.automation.APIServer.APIServer.APIServer
                  method), 664
                                                                                                                                 method), 16, 92, 149
add_traits()(AFL.automation.prepare.SweepBuilderWidkFL.Taxttomation
                  method), 674
                                                                                                                        module, 13, 789
add_traits() (AFL.automation.prepare.SweepBuilderWidkFLVBoxtomation.APIServer
                                                                                                                        module, 14, 83
                  method), 681
                                                                                                              AFL.automation.APIServer.APIServer
add_unqueued_routes()
                  (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                        module, 14, 83
                  method), 16, 92, 149
                                                                                                               AFL.automation.APIServer.Client
```

module, 17, 151	module, 38, 256
AFL.automation.APIServer.Driver	AFL.automation.loading.PneumaticPressureSampleCell
module, 19, 156	module, 39, 270
AFL.automation.APIServer.DummyDriver	AFL.automation.loading.PneumaticSampleCell
module, 20, 163	module, 41, 280
AFL.automation.APIServer.DummyOT2Driver	AFL.automation.loading.PressureController
module, 21, 170	module, 42, 289
AFL.automation.APIServer.LoggerFilter	AFL.automation.loading.PressureControllerAsPump
module, 22, 176	module, 42, 291
AFL.automation.APIServer.QueueDaemon	AFL.automation.loading.PushPullSelectorSampleCell
module, 22, 176	module, 43, 294
AFL.automation.EpicsADLiveProcess	AFL.automation.loading.RSoXSSolutionSampleCell
module, 23, 790	module, 45, 305
AFL.automation.EpicsADLiveProcess.AreaDetecto	
module, 23, 790	module, 47, 316
AFL.automation.EpicsADLiveProcess.Client	AFL.automation.loading.SampleCell
module, 24, 791	module, 48, 320
AFL.automation.EpicsADLiveProcess.ReduceDaemo	
module, 24, 793	module, 48, 321
AFL.automation.instrument	AFL.automation.loading.SensorCallbackThread
module, 25, 183	module, 50, 328
AFL.automation.instrument.DummySAS	AFL.automation.loading.SensorPollingThread
module, 25, 183	module, 52, 343
AFL.automation.instrument.FileCamera	AFL.automation.loading.SerialDevice
module, 26, 188	module, 53, 347
AFL.automation.instrument.I22SAXS	AFL.automation.loading.SyringePump
module, 26, 188	module, 53, 348
AFL.automation.instrument.NetworkCamera	AFL.automation.loading.Tubing
module, 27, 193	module, 54, 349
AFL.automation.instrument.SeabreezeUVVis	AFL.automation.loading.TwoSelectorBlowoutSampleCell
module, 27, 194	module, 54, 350
AFL.automation.instrument.SpecScreen_Driver	AFL.automation.loading.UltimusVPressureController
module, 28, 207	module, 56, 361
AFL.automation.loading	AFL.automation.loading.ViciMultiposSelector
module, 29, 213	module, 57, 364
AFL.automation.loading.CetoniMultiPosValve	AFL.automation.prepare
module, 31, 215	module, 57, 367
AFL.automation.loading.ChemyxSyringePump	AFL.automation.prepare.Component
module, 31, 216	module, 58, 380
AFL.automation.loading.DigitalOutPressureCont	rMoHILeautomation.prepare.DeckBuilderWidget
module, 33, 223	module, 59, 384
AFL.automation.loading.DoubleViciMultiposSele	cABGr.automation.prepare.Dummy_OT2_Driver
module, 34, 225	module, 61, 455
AFL.automation.loading.DummyPump	AFL.automation.prepare.factory
module, 35, 229	module, 68, 687
AFL.automation.loading.FlowSelector	AFL.automation.prepare.OT2Client
module, 35, 232	module, 62, 463
AFL.automation.loading.LoadStopperDriver	AFL.automation.prepare.PrepareWidget
module, 35, 232	module, 63, 473
AFL.automation.loading.MultiChannelRelay	AFL.automation.prepare.PrepType
module, 36, 251	module, 63, 470
AFL.automation.loading.NE1kSyringePump	AFL.automation.prepare.SampleSeriesWidget
module, 37, 252	module, 64, 545
AFL.automation.loading.OneSelectorBlowoutSamp	
III I . aa comacton . toaarng . onesetee corbrowou coamp	

attribute), 598

module, 66, 625	$\verb"align_content" (AFL. automation. prepare. Sweep Builder Widget. Layout$
AFL.automation.prepare.SweepBuilderWidget	attribute), 662
module, 67, 628	$\verb"align_items" (AFL. automation. prepare. Deck Builder Widget. Layout$
AFL.automation.prepare.utilities	attribute), 432
module, 69, 692	$\verb"align_items" (AFL. automation. prepare. Prepare Widget. Layout$
AFL.automation.sample	attribute), 517
module, 82, 692	$\verb"align_items" (AFL. automation. prepare. Sample Series Widget. Layout$
AFL.automation.sample.CastingServer	attribute), 598
module, 82, 693	$\verb"align_items" (AFL. automation. prepare. Sweep Builder Widget. Layout$
AFL.automation.sample_env	attribute), 662
module, 69, 705	$\verb align_self (AFL. automation. prepare. Deck Builder Widget. Layout$
AFL.automation.sample_env.TemperatureDeck	attribute), 432
module, 69, 705	$\verb align_self (AFL. automation. prepare. Prepare Widget. Layout$
AFL.automation.shared	attribute), 517
module, 70, 710	$\verb align_self (AFL. automation. prepare. Sample Series Widget. Layout$
AFL.automation.shared.DataLabelerWidget	attribute), 598
module, 70, 711	$\verb"align_self" (AFL. automation. prepare. Sweep Builder Widget. Layout$
AFL.automation.shared.DatasetWidget	attribute), 662
module, 72, 722	$\verb"alive()" (AFL. automation. loading. Load Stopper Driver. Sensor Polling Three distances of the property of$
AFL.automation.shared.DiffractionLabeler	method), 243
module, 74, 732	alive() (AFL.automation.loading.SensorPollingThread.SensorPollingTh
AFL.automation.shared.exceptions	method), 53, 345, 347
module, 79, 780	All (AFL.automation.APIServer.APIServer.IPVersion at-
AFL.automation.shared.MutableQueue	tribute), 127
module, 75, 742	All (AFL.automation.shared.ServerDiscovery.IPVersion
AFL.automation.shared.PersistentConfig	attribute), 760
module, 76, 743	analyze_stocks_cb()
AFL.automation.shared.serialization	(AFL.automation.prepare.PrepareWidget.StockBuilderWidget
module, 80, 781	method), 528
AFL.automation.shared.ServerDiscovery	analyze_stocks_cb()
module, 77, 748	(AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget)
AFL.automation.shared.units	method), 66, 626, 627
module, 80, 782	anchor (AFL.automation.instrument.SeabreezeUVVis.Path
AFL.automation.shared.utilities	property), 202
module, 81, 783	APIServer (class in AFL.automation.APIServer.APIServer),
AFL.automation.shared.widgetui	15, 90, 149
module, 81, 784	app (AFL.automation.loading.LoadStopperDriver.LoadStopperDriver
after_request() (AFL.automation.APIServer.APISer	
method), 119	app (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSelecto
after_request_funcs	property), 261
(AFL.automation.APIServer.APIServer.Flask	app (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelecto
attribute), 123	property), 265
aio_find_server_by_name()	app (AFL.automation.loading.PneumaticPressureSampleCell.Pn
(AFL.automation.APIServer.Client.ServerDisc	
method), 154	app (AFL.automation.loading.PneumaticSampleCell.PneumaticSampleCell
aio_find_server_by_name()	property), 41, 285, 289
	verD issc (AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelec
method), 79, 764, 779	property), 44, 299, 304
	Wid gppLAN Flutuutomation.loading.RSoXSSolutionSampleCell.RSoXSSolutionS
attribute), 432	property), 46, 310, 315
	get.Iapp(uAFL.automation.loading.TwoSelectorBlowoutSampleCell.TwoSelector
attribute), 517	property), 55, 356, 361
	sWi dper_Lagrotie xt() (AFL.automation.APIServer.APIServer.Flask
	w. =/

810 Index

method), 125

app_ctx_globals_class	method), 778
(AFL.automation.APIServer.APIServer.Flask	async_clear_cache()
attribute), 104	(AFL. automation. API Server. API Server. Service Info
${\tt apply_isel()} \ (AFL. automation. shared. Dataset Widget. Dataset. Dataset.$	DatasetWidg en ethod), 142
method), 73, 725, 731	async_clear_cache()
	DatasetWidg &<u>A</u>Madul omation.shared.ServerDiscovery.AsyncServiceInfo
method), 73, 727, 731	method), 754
apply_protocol_order()	async_clear_cache()
(AFL.automation.prepare.PrepareWidget.Sample method), 526	leSeriesWidg&FL.automation.shared.ServerDiscovery.ServiceInfo method), 771
<pre>apply_protocol_order()</pre>	<pre>async_close() (AFL.automation.shared.ServerDiscovery.AsyncZerocor</pre>
(AFL. automation. prepare. Sample Series Widget. Some anti-control of the property of the pr	SampleSerie xWeitlgel t), 758
method), 65, 606, 624	$\verb"async_find()" (AFL. automation. shared. Server Discovery. A sync Zero confidence of the property of the pr$
apply_protocol_order_cb()	class method), 759
(AFL. automation. prepare. Prepare Widget. Sample	
method), 526	(AFL. automation. API Server. API Server. Zero conf
apply_protocol_order_cb()	method), 147
(AFL. automation. prepare. Sample Series Widget. Some state of the support of t	
method), 65, 606, 624	(AFL. automation. shared. Server Discovery. A sync Zero conful for the property of the prope
${\tt apply_sel()} \ (AFL. automation. shared. Dataset Widget. Dataset. Dataset$	
method), 73, 725, 731	<pre>async_get_service_info()</pre>
	atasetWidget _MFddal utomation.shared.ServerDiscovery.Zeroconf
method), 73, 726, 731	method), 777
	$\verb async_notify_all() (AFL. automation. API Server. API Server. Zero configuration and approximation of the property of the $
AFL.automation.EpicsADLiveProcess.AreaDetec	
23, 790, 791	async_notify_all() (AFL.automation.shared.ServerDiscovery.Zeroco.
args (AFL. automation. prepare. Component. Parse Exception Arguments and Arguments a	
attribute), 383	async_register_service()
as_posix() (AFL.automation.instrument.SeabreezeUVVis method), 202	is.Path (AFL.automation.APIServer.APIServer.Zeroconf method), 147
as_uri()(AFL.automation.instrument.SeabreezeUVVis.Pa	Padsync_register_service()
method), 202	(AFL. automation. shared. Server Discovery. A sync Zero conful and the property of the prope
$aspirate_rate()$ (AFL.automation.prepare.OT2Client.O	OT2Client method), 757
method), 62, 468, 470	async_register_service()
<pre>aspirate_rate() (AFL.automation.sample.CastingServe</pre>	ver.OT2Clien(AFL.automation.shared.ServerDiscovery.Zeroconf method), 777
assign_targets() (AFL.automation.sample.CastingServ	
	(AFL.automation.shared.ServerDiscovery.AsyncZeroconf
<pre>async_add_listener()</pre>	method), 758
(AFL.automation.APIServer.APIServer.Zeroconf	
method), 148	(AFL.automation.APIServer.APIServer.Zeroconf
async_add_listener()	method), 148
(AFL.automation.shared.ServerDiscovery.Zeroco	confsync_remove_listener()
method), 778	(AFL.automation.shared.ServerDiscovery.Zeroconf
<pre>async_add_service_listener()</pre>	method), 778
(AFL.automation.shared.ServerDiscovery.Async2	cZ arsymo fremove_service_listener()
method), 758	(AFL. automation. shared. Server Discovery. A sync Zero conful and the property of the prope
<pre>async_cancel() (AFL.automation.shared.ServerDiscover</pre>	ery.AsyncSe miathBd dyv858
method), 751	$async_request()$ (AFL.automation.APIServer.APIServer.ServiceInfo
<pre>async_check_service()</pre>	method), 142
	$f \ \ {\tt async_request()} \ (AFL. automation. shared. Server Discovery. Async Servi$
method), 148	method), 754
<pre>async_check_service()</pre>	$\verb async_request() (AFL. automation. shared. Server Discovery. Service Information and Server Discovery. Service Information and Server Discovery Discover$
(AFL.automation.shared.ServerDiscovery.Zeroco	conf method), 771

$\verb"async_send()" (AFL. automation. API Server. API Server. Zerver. API Server. Zerver. API Server. AP$	<i>∵oarsoynj</i> hc_u		
method), 149		(AFL. automation. API Server. API Server. Zero	conf
$\verb"async_send()" (AFL. automation. shared. Server Discovery. Zerver Discovery. Zerv$			
method), 778	-	pdate_service()	
$\verb"async_to_sync()" (AFL. automation. APIS erver. API$:Flask	(AFL.automation.shared.ServerDiscovery.A.	syncZeroconf
method), 117		method), 758	
<pre>async_unregister_all_services()</pre>	-	pdate_service()	
(AFL.automation.APIServer.APIServer.Zeroconf		(AFL.automation.shared.ServerDiscovery.Ze	eroconf
method), 148		method), 777	
<pre>async_unregister_all_services()</pre>	-	<pre>ait() (AFL.automation.APIServer.APIServe</pre>	r.ServiceInfo
(AFL.automation.shared.ServerDiscovery.Async2			
method), 757	async_w	ait()(AFL.automation.APIServer.APIServe	r.Zeroconf
<pre>async_unregister_all_services()</pre>		method), 146	
(AFL. automation. shared. Server Discovery. Zerocomulation and the property of the property	mafsync_w		very.AsyncServiceInfo
method), 778		method), 754	
<pre>async_unregister_service()</pre>		<pre>ait() (AFL.automation.shared.ServerDiscov</pre>	very.ServiceInfo
(AFL.automation.APIServer.APIServer.Zeroconf		method), 771	
method), 148	async_w	<pre>ait() (AFL.automation.shared.ServerDiscov</pre>	very.Zeroconf
<pre>async_unregister_service()</pre>		method), 776	
(AFL.automation.shared.ServerDiscovery.Async2	Z avsymo £w		
method), 758		(AFL. automation. API Server. API Server. Zero	conf
<pre>async_unregister_service()</pre>		method), 146	
(AFL.automation.shared.ServerDiscovery.Zeroco	ongsync_w		
method), 777		(AFL. automation. shared. Server Discovery. Zerver Discovery. Ze	eroconf
<pre>async_update_records()</pre>		method), 776	
(AFL.automation.APIServer.APIServer.ServiceIn	f A syncSe	rviceBrowser (class	in
method), 142		AFL.automation.shared.ServerDiscovery),	
<pre>async_update_records()</pre>		749	
(AFL.automation.shared.ServerDiscovery.Async	SeksyjandSve	www.ceInfo (class	in
method), 751		AFL.automation.shared.ServerDiscovery),	
<pre>async_update_records()</pre>		751	
(AFL.automation.shared.ServerDiscovery.AsyncS	Seksyjoned Zi _l fa		in
method), 754		AFL.automation.shared.ServerDiscovery),	
<pre>async_update_records()</pre>		756	
(AFL.automation.shared.ServerDiscovery.Service	e Assynn.ce Zre		in
method), 766		AFL.automation.shared.ServerDiscovery),	
<pre>async_update_records()</pre>		758	
(AFL. automation. shared. Server Discovery. Service and the state of			'3
method), 771	auto_fi	nd_instance_path()	
<pre>async_update_records_complete()</pre>		(AFL.automation.APIServer.APIServer.Flast	k
(AFL.automation.APIServer.APIServer.ServiceIn	fo	method), 108	
method), 142	D		
<pre>async_update_records_complete()</pre>	В		
(AFL.automation.shared.ServerDiscovery.Async	Swaierre	Wiff&\$s() (AFL.automation.prepare.MassBalo	ance
method), 751		method), 372	
<pre>async_update_records_complete()</pre>		r (AFL.automation.prepare.Sample proper	ty),
(AFL.automation.shared.ServerDiscovery.AsyncS	ServiceInfo	2373	
method), 754	BaseCom	ponent (AFL.automation.prepare.Componer	ıt.PrepType
<pre>async_update_records_complete()</pre>		attribute), 382	
(AFL.automation.shared.ServerDiscovery.Service	e Besecc m	ponent (AFL.automation.prepare.PrepType	РгерТуре
method), 767		attribute), 63, 472, 473	= -=
<pre>async_update_records_complete()</pre>	BaseMix	ture (AFL.automation.prepare.Component.F	РгерТуре
(AFL.automation.shared.ServerDiscovery.Service method), 771	e I nfo	attribute), 382	

BaseMixture (AFL.automation.prepare.PrepType.PrepTypattribute), 63, 472, 473	ppe (AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSe method), 39, 261, 269
<pre>before_first_request()</pre>	blowOutCellLegacy()
(AFL.automation.APIServer.APIServer.Flask method), 114	(AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSe method), 266
before_first_request_funcs	blowOutCellLegacy()
(AFL.automation.APIServer.APIServer.Flask attribute), 106	(AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolut method), 46, 311, 316
before_request() (AFL.automation.APIServer.API	ve bEbw@ utCellLegacy()
method), 119	(AFL. automation. loading. Two Selector Blow out Sample Cell. Two Selector Blow out Selector Blow out Sample Cell. Two Selector Blow out Selector Blow
before_request_funcs	method), 56, 357, 361
(AFL.automation.APIServer.APIServer.Flask attribute), 123	blueprints (AFL.automation.APIServer.APIServer.Flask attribute), 106
blockUntilStatusStopped()	blur() (AFL.automation.prepare.DeckBuilderWidget.Button
(AFL.automation.loading.ChemyxSyringePump.	
method), 32, 221, 222	blur() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
blockUntilStatusStopped()	method), 395
= = · · ·	on biblie f. Distilial. Qui d'massi une prepard l'Deck Builder Widget. Dropdown
method), 224	method), 408
blockUntilStatusStopped()	blur() (AFL.automation.prepare.DeckBuilderWidget.HBox
(AFL.automation.loading.DigitalOutPressureCo.	
method), 225	blur() (AFL.automation.prepare.DeckBuilderWidget.Label
blockUntilStatusStopped()	method), 424
	Pholopr() (AFL.automation.prepare.DeckBuilderWidget.Text
method), 35, 230, 231	method), 442
blockUntilStatusStopped()	blur() (AFL.automation.prepare.DeckBuilderWidget.VBox
(AFL.automation.loading.NE1kSyringePump.NE	
method), 38, 254, 255	blur() (AFL.automation.prepare.PrepareWidget.Button
blockUntilStatusStopped()	method), 477
	res is live(Q)(tAdHen utomation.prepare.PrepareWidget.Checkbox
method), 42, 290	method), 484
blockUntilStatusStopped()	blur() (AFL.automation.prepare.PrepareWidget.Dropdown
(AFL.automation.loading.PressureControllerAsI	
method), 43, 292, 294	blur() (AFL.automation.prepare.PrepareWidget.HBox
blockUntilStatusStopped()	method), 501
(AFL.automation.loading.UltimusVPressureCon	ntr bl/ar.P) rest AlEC.antrollæt ion.prepare.PrepareWidget.Label
method), 362	method), 509
blockUntilStatusStopped()	blur() (AFL.automation.prepare.PrepareWidget.Text
(AFL.automation.loading.UltimusVPressureCon	
method), 363	blur() (AFL.automation.prepare.PrepareWidget.VBox
blowOutCell() (AFL.automation.loading.OneSelectorBlo	
method), 39, 261, 269	blur() (AFL.automation.prepare.SampleSeriesWidget.Button
blowOutCell() (AFL.automation.loading.OneSelectorBlo	
method), 266	blur() (AFL.automation.prepare.SampleSeriesWidget.Checkbox
blowOutCell() (AFL.automation.loading.PushPullSelect	
method), 45, 301, 305	blur() (AFL.automation.prepare.SampleSeriesWidget.FloatText
blowOutCell() (AFL.automation.loading.RSoXSSolution	
method), 46, 311, 316	blur() (AFL.automation.prepare.SampleSeriesWidget.HBox
blowOutCell() (AFL.automation.loading.TwoSelectorBlo	
method), 56, 357, 361	blur() (AFL.automation.prepare.SampleSeriesWidget.IntText
blowOutCellForcedAir()	method), 582
	pl &Cvlt.PusAPIdlSalvonatiSapplepGvl lSampleSeriesWidget.Label
method), 45, 301, 305	method), 590
blowOutCellLegacy()	$\verb blur() (AFL. automation. prepare. Sample Series Widget. Text $

```
method), 611
                                                                                                                                                                                                 attribute), 662
blur() (AFL.automation.prepare.SampleSeriesWidget.VBobottom(AFL.automation.prepare.DeckBuilderWidget.Layout
                           method), 618
                                                                                                                                                                                                 attribute), 432
blur() (AFL.automation.prepare.SweepBuilderWidget.Buttbottom (AFL.automation.prepare.PrepareWidget.Layout
                           method), 631
                                                                                                                                                                                                 attribute), 517
blur() (AFL.automation.prepare.SweepBuilderWidget.Chebdatxom (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                                                                                                                                                                 attribute), 599
                           method), 639
\verb|blur()| (AFL. automation. prepare. SweepBuilder Widget. HBd \verb|bottom| (AFL. automation. prepare. SweepBuilder Widget. Layout) | (AFL. automation. prepare. Pre
                           method), 647
                                                                                                                                                                                                 attribute), 663
\verb|blur()| (AFL. automation. prepare. SweepBuilder Widget. Labbox\_style (AFL. automation. prepare. DeckBuilder Widget. HBox) | (AFL. automation. HBox) | 
                           method), 654
                                                                                                                                                                                                 attribute), 416
blur() (AFL.automation.prepare.SweepBuilderWidget.Textbox_style (AFL.automation.prepare.DeckBuilderWidget.VBox
                           method), 674
                                                                                                                                                                                                 attribute), 449
blur() (AFL.automation.prepare.SweepBuilderWidget.VBobox_style(AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                                                                                 attribute), 501
                           method), 681
border (AFL.automation.prepare.DeckBuilderWidget.Layolbox_style (AFL.automation.prepare.PrepareWidget.VBox
                           property), 434
                                                                                                                                                                                                 attribute), 539
border (AFL.automation.prepare.PrepareWidget.Layout box_style(AFL.automation.prepare.SampleSeriesWidget.HBox
                           property), 519
                                                                                                                                                                                                 attribute), 575
border (AFL.automation.prepare.SampleSeriesWidget.Laydbax_style (AFL.automation.prepare.SampleSeriesWidget.VBox
                           property), 600
                                                                                                                                                                                                 attribute), 618
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), 647
                           property), 664
border_bottom(AFL.automation.prepare.DeckBuilderWidgetLStonlte(AFL.automation.prepare.SweepBuilderWidget.VBox
                           attribute), 432
                                                                                                                                                                                                 attribute), 681
border_bottom(AFL.automation.prepare.PrepareWidget.lbuyiolad_deck_object()
                           attribute), 517
                                                                                                                                                                                                 (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget
border_bottom(AFL.automation.prepare.SampleSeriesWidget.Layoutethod), 60, 404, 455
                                                                                                                                                                     build_deck_object()
                           attribute), 598
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), 662
                                                                                                                                                                                                 method), 60, 405, 455
border_left (AFL.automation.prepare.DeckBuilderWidgebflatledutdeck_object()
                           attribute), 432
                                                                                                                                                                                                 (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget
border_left (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                                                                                 method), 490
                           attribute), 517
                                                                                                                                                                     build_label() (AFL.automation.prepare.PrepareWidget.SampleSeriesWi
border_left (AFL.automation.prepare.SampleSeriesWidget.Layout method), 526
                           attribute), 599
                                                                                                                                                                     build_label() (AFL.automation.prepare.SampleSeriesWidget.SampleSer
border_left (AFL.automation.prepare.SweepBuilderWidget.Layout method), 65, 606, 624
                            attribute), 663
                                                                                                                                                                     bulk_cast_films() (AFL.automation.sample.CastingServer.CastingServ
border_right (AFL.automation.prepare.DeckBuilderWidget.Layout method), 82, 695, 705
                                                                                                                                                                     Button (class in AFL.automation.prepare.DeckBuilderWidget),
                           attribute), 432
border_right (AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                                                     Button (class in AFL.automation.prepare.PrepareWidget),
                           attribute), 517
border_right (AFL.automation.prepare.SampleSeriesWidget.Layout474
                                                                                                                                                                     Button (class in AFL.automation.prepare.SampleSeriesWidget),
                           attribute), 598
border_right (AFL.automation.prepare.SweepBuilderWidget.Layou546
                                                                                                                                                                     Button (class in AFL.automation.prepare.SweepBuilderWidget),
                           attribute), 662
border_top(AFL.automation.prepare.DeckBuilderWidget.Layout
                                                                                                                                                                                               629
                           attribute), 432
                                                                                                                                                                     \verb|button_style| (AFL. automation. prepare. Deck Builder Widget. Button
border_top(AFL.automation.prepare.PrepareWidget.Layout
                                                                                                                                                                                                 attribute), 387
                                                                                                                                                                     \verb|button_style| (AFL. automation. prepare. Prepare Widget. Button
                           attribute), 517
border_top(AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 476
                           attribute), 598
                                                                                                                                                                     \verb|button_style| (AFL. automation. prepare. Sample Series Widget. Button
border_top (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 548
```

```
button_style (AFL.automation.prepare.SweepBuilderWidgbffBnttb) (AFL.automation.EpicsADLiveProcess.AreaDetectorLive.AreaD
                                                   attribute), 631
                                                                                                                                                                                                                                                                                                                                                                method), 24, 791
                                                                                                                                                                                                                                                                                                             ceil() (in module AFL.automation.APIServer.Driver),
C
calc_sweep() (AFL.automation.prepare.SweepBuilderWidgei.sweepBunderWidgerangometion.APIServer.DummyDriver),
                                                   method), 68, 671, 687
\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), 529
calc_sweep_cb() (AFL.automation.prepare.SweepBuilder \\ \frac{\partials}{\partials} \) \( \text{SweepBuilty} \) \( \text{
                                                  method), 67, 670, 687
\verb|calculate_bounds()| (AFL. automation. prepare. Mass Bala \textit{Rei} 1() (in module AFL. automation. sample. Casting Server), \\
                                                 method), 372
                                                                                                                                                                                                                                                                                                      cellToWaste() (AFL.automation.loading.OneSelectorBlowoutSampleCell
calibrate() (AFL.automation.loading.Sensor.Sensor
                                                                                                                                                                                                                                                                                                                                                                method), 261
                                                   method), 49, 327
calibrate_sensor() (AFL.automation.loading.LoadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadStop6elDrivelestadS
                                                                                                                                                                                                                                                                                                                                                                method), 266
                                                  method), 36, 240, 250
calibrate_sensor() (AFL.automation.loading.PneumaticPlesTaWasangleCell.PnetamaticPlesadiesangleCellSelectorSampleCell.Pi
                                                                                                                                                                                                                                                                                                                                                               method), 45, 299, 305
                                                   method), 40, 275, 280
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), 46, 310, 316
                                                 method), 42, 285, 289
camera_reset() (AFL.automation.instrument.NetworkCameralNeWasteComeraL.automation.loading.TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                                                                               method), 55, 356, 361
                                                   method), 27, 194
\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), 767
                                                                                                                                                                                                                                                                                                                                                               31, 215, 216
CastingServer
                                                                                                                                                                        (class
                                                                                                                                                                                                                                                                                       in
                                                                                                                                                                                                                                                                                                             change_model_callback()
                                                 AFL.automation.sample.CastingServer),
                                                                                                                                                                                                                                                                                                                                                                (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
                                                  82, 693, 705
                                                                                                                                                                                                                                                                                                                                                               method), 71, 715, 721
catch_sample()
                                                                                                                                   (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                             change_model_callback()
                                                   method), 370
catchToSyringe() (AFL.automation.loading.OneSelectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSafafleCutt.OntiselectorBlowoutSaf
                                                                                                                                                                                                                                                                                                                                                                method), 75, 733, 741
                                                  method), 261
catchToSyringe() (AFL.automation.loading.OneSelectorBlowsun SunfpleCell? Word State of the Color of the Color
                                                                                                                                                                                                                                                                                                                                                               (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
                                                   method), 266
catchToSyringe() (AFL.automation.loading.PushPullSelectorSampleCellPushPullSelectorSampleCell
                                                                                                                                                                                                                                                                                                              change_norder_callback()
                                                  method), 44, 299, 304
catchToSyringe() (AFL.automation.loading.RSoXSSolutionSample(ATLRSUNSSOLUTION) in the Control of the Control of
                                                                                                                                                                                                                                                                                                                                                               method), 75, 733, 741
                                                   method), 46, 310, 316
catchToSyringe() (AFL.automation.loading.TwoSelectorBlowGasSantpACCEN.ThoseleCtorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                                                                               (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
                                                  method), 55, 356, 361
catchToWaste() (AFL.automation.loading.OneSelectorBlowoutSample@ed.OneSelectorBlowoutSampleCell
                                                                                                                                                                                                                                                                                                             change_qstar_callback()
                                                  method), 261
catchToWaste() (AFL.automation.loading.OneSelectorBlowoutSampleCell!#wwstientshBted.DiffsantjienCellpeler.DiffractionLabeler
                                                                                                                                                                                                                                                                                                                                                                method), 75, 733, 741
                                                  method), 266
catchToWaste() (AFL.automation.loading.PushPullSelect& Participal Controller.Ulti
                                                                                                                                                                                                                                                                                                                                                                method), 56, 363, 364
                                                   method), 44, 299, 304
catchToWaste() (AFL.automation.loading.RSoXSSolutionShafelectellpressessolulatellsautomation.APIServer.APIServer.QueueDaemo
                                                                                                                                                                                                                                                                                                                                                               method), 135
                                                   method), 46, 310, 316
catchToWaste() (AFL.automation.loading.TwoSelectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwoselectorBlothensairfupeunstwosel
                                                                                                                                                                                                                                                                                                                                                                method), 23, 180, 182
                                                  method), 55, 356, 361
{\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), 716
categories_(AFL.automation.shared.DiffractionLabeler.Ohminkher.Caless in AFL.automation.prepare.PrepareWidget),
                                                  attribute), 736
                                                                                                                                                                                                                                                                                                             Checkbox (class in AFL.automation.prepare.SampleSeriesWidget),
```

	ss_own_trait_events()
Checkbox (class in AFL.automation.prepare.SweepBuilderWidg 637	**
ChemyxConnection (class in cla	ss_own_trait_events()
AFL.automation.loading.ChemyxSyringePump), 32, 217, 222	(AFL.automation.prepare.PrepareWidget.Dropdown class method), 493
ChemyxSyringePump (class in cla	ss_own_trait_events()
AFL.automation.loading.ChemyxSyringePump), 32, 219, 221	(AFL.automation.prepare.PrepareWidget.HBox class method), 501
children (AFL.automation.prepare.DeckBuilderWidget.HBoha	ss_own_trait_events()
attribute), 416 children (AFL.automation.prepare.DeckBuilderWidget.VBox	(AFL.automation.prepare.PrepareWidget.Label class method), 509
	ss_own_trait_events()
children (AFL.automation.prepare.PrepareWidget.HBox attribute), 501	(AFL.automation.prepare.PrepareWidget.Layout class method), 519
children (AFL.automation.prepare.PrepareWidget.VBox cla	ss_own_trait_events()
attribute), 540	(AFL.automation.prepare.PrepareWidget.Text
${\tt children} \ (AFL. automation. prepare. Sample Series Widget. HBox$	
attribute), 575 cla	ss_own_trait_events()
children (AFL.automation.prepare.SampleSeriesWidget.VBox attribute), 619	(AFL.automation.prepare.PrepareWidget.VBox class method), 540
children (AFL.automation.prepare.SweepBuilderWidget.HBka	
attribute), 647 children (AFL.automation.prepare.SweepBuilderWidget.VBox	(AFL. automation. prepare. Sample Series Widget. Button
	ss_own_trait_events()
chmod() (AFL.automation.instrument.SeabreezeUVVis.Path method), 200	(AFL.automation.prepare.SampleSeriesWidget.Checkbox class method), 556
CIASS_OWIL_CIAIC_EVELICS()	SS_OWN_trait_events()
(AFL. automation. prepare. Deck Builder Widget. Button	ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events()
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events()
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() class_own_trait_events((AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 424	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Text
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 611
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Text class method), 442	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 SS_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 SS_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 SS_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 SS_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600 SS_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 611 SS_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.VBox
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Text class method), 442 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Text class method), 442 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Text class method), 442 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.VBox	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 611 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.VBox class method), 619 ss_own_trait_events() (AFL.automation.prepare.SweepBuilderWidget.Button
(AFL.automation.prepare.DeckBuilderWidget.Button class method), 388 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Checkb class method), 395 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Dropdo class method), 408 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.HBox class method), 416 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 424 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Label class method), 434 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 434 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.Text class method), 442 class_own_trait_events() cla (AFL.automation.prepare.DeckBuilderWidget.VBox class method), 450	(AFL.automation.prepare.SampleSeriesWidget.FloatText class method), 567 ss_own_trait_events() ox (AFL.automation.prepare.SampleSeriesWidget.HBox class method), 575 ss_own_trait_events() wn (AFL.automation.prepare.SampleSeriesWidget.IntText class method), 582 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Label class method), 590 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Layout class method), 600 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 611 ss_own_trait_events() (AFL.automation.prepare.SampleSeriesWidget.VBox class method), 619 ss_own_trait_events()

<pre>class_own_trait_events()</pre>	class method), 575
(AFL. automation. prepare. Sweep Builder World States and States and States are supported by the states are supp	${\it Vidget.HR}$ drass_own_traits() (AFL. automation. prepare. Sample Series Widget. Int T
class method), 647	class method), 583
<pre>class_own_trait_events()</pre>	$\verb class_own_traits() (AFL. automation. prepare. Sample Series Widget. Lab \\$
(AFL. automation. prepare. Sweep Builder World States and States and States are supported by the states are supp	
class method), 654	$\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 $
<pre>class_own_trait_events()</pre>	class method), 601
	lidget.La odas s_own_traits()(AFL.automation.prepare.SampleSeriesWidget.Text
class method), 664	class method), 611
<pre>class_own_trait_events()</pre>	$\verb class_own_traits() (AFL. automation. prepare. Sample Series Widget. VBound for the property of the prope$
(AFL. automation. prepare. Sweep Builder World States and States are the support of the states are the support of the states are the states	
class method), 674	$\verb class_own_traits() (AFL. automation. prepare. SweepBuilder Widget. But $
<pre>class_own_trait_events()</pre>	class method), 631
	$\it Vidget.VB$ chass $\it countraits$ () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare . Sweep Builder Widget. Characteristics () (AFL automation . prepare .
class method), 682	class method), 639
	eckBuild er&EslgowButtoa its() (AFL.automation.prepare.SweepBuilderWidget.HB
class method), 388	class method), 647
	eckBuild er&sislgow6]terklic as() (AFL.automation.prepare.SweepBuilderWidget.Lab
class method), 395	class method), 654
	DeckBuild er&Est[gowD_topdows () (AFL.automation.prepare.SweepBuilderWidget.Lay
class method), 408	class method), 665
	eckBuild er&sislgowHBcxa its() (AFL.automation.prepare.SweepBuilderWidget.Text
class method), 416	class method), 674
class_own_traits()(AFL.automation.prepare.D	eckBuild er&sislgowhaharla its() (AFL.automation.prepare.SweepBuilderWidget.VBo
class method), 424	class method), 682
${\tt class_own_traits()} \ (AFL. automation. prepare. D$	
class method), 434	(AFL. automation. prepare. Deck Builder Widget. Button
${\tt class_own_traits()} \ (AFL. automation. prepare. D$	
class method), 442	<pre>class_trait_names()</pre>
	eckBuilderWidget(AB dxautomation.prepare.DeckBuilderWidget.Checkbox
class method), 450	class method), 396
${\tt class_own_traits()} \ (AFL. automation. prepare. P$	
class method), 477	(AFL. automation. prepare. Deck Builder Widget. Drop down
${\tt class_own_traits()} \ (AFL. automation. prepare. P$	
class method), 484	<pre>class_trait_names()</pre>
class_own_traits()(AFL.automation.prepare.P	repareWidget.DropAbkmutomation.prepare.DeckBuilderWidget.HBox
class method), 493	class method), 417
${\tt class_own_traits()} \ (AFL. automation. prepare. P$	repareWill@ssHRvait_names()
class method), 501	(AFL. automation. prepare. Deck Builder Widget. Label
${\tt class_own_traits()} \ (AFL. automation. prepare. P$	repareWidget.Lab el ass method), 424
class method), 509	<pre>class_trait_names()</pre>
${\tt class_own_traits()}\ (AFL. automation. prepare. P$	repareWidget.Lay@AFL.automation.prepare.DeckBuilderWidget.Layout
class method), 519	class method), 434
${\tt class_own_traits()}\ (AFL. automation. prepare. P$	repareWid ges Scurait_names()
class method), 532	(AFL.automation.prepare.DeckBuilderWidget.Text
<pre>class_own_traits() (AFL.automation.prepare.P</pre>	repareWidget.VBoxlass method), 442
class method), 540	<pre>class_trait_names()</pre>
<pre>class_own_traits() (AFL.automation.prepare.Set</pre>	ampleSeriesWidge tABiltanu tomation.prepare.DeckBuilderWidget.VBox
class method), 549	class method), 450
<pre>class_own_traits() (AFL.automation.prepare.Sa</pre>	ampleSe cilea BisdyutaGheadanna s()
class method), 556	(AFL.automation.prepare.PrepareWidget.Button
<pre>class_own_traits() (AFL.automation.prepare.Set</pre>	
class method), 567	<pre>class_trait_names()</pre>
<pre>class_own_traits() (AFL.automation.prepare.Set</pre>	ampleSeriesWidge(AffLonutomation.prepare.PrepareWidget.Checkbox

class method), 485	class method), 647
<pre>class_trait_names()</pre>	<pre>class_trait_names()</pre>
(AFL.automation.prepare.PrepareWidget.Dropdo	own (AFL.automation.prepare.SweepBuilderWidget.Label
class method), 493	class method), 655
<pre>class_trait_names()</pre>	<pre>class_trait_names()</pre>
(AFL.automation.prepare.PrepareWidget.HBox	(AFL.automation.prepare.SweepBuilderWidget.Layout
class method), 502	class method), 665
class_trait_names()	<pre>class_trait_names()</pre>
(AFL.automation.prepare.PrepareWidget.Label	(AFL.automation.prepare.SweepBuilderWidget.Text
class method), 509	class method), 674
class_trait_names()	class_trait_names()
(AFL.automation.prepare.PrepareWidget.Layout	
class method), 519	class method), 682
<pre>class_trait_names()</pre>	class_traits()(AFL.automation.prepare.DeckBuilderWidget.Button
(AFL.automation.prepare.PrepareWidget.Text	class method), 388
class method), 532	class_traits()(AFL.automation.prepare.DeckBuilderWidget.Checkbo.
class_trait_names()	class method), 396
(AFL.automation.prepare.PrepareWidget.VBox	
class method), 540	class method), 409
class_trait_names()	class_traits()(AFL.automation.prepare.DeckBuilderWidget.HBox
(AFL.automation.prepare.SampleSeriesWidget.B	
class method), 549	class_traits()(AFL.automation.prepare.DeckBuilderWidget.Label
class_trait_names()	class method), 424
	heddss_traits()(AFL.automation.prepare.DeckBuilderWidget.Layout
class method), 557	class method), 434
class_trait_names()	class_traits() (AFL.automation.prepare.DeckBuilderWidget.Text
(AFL.automation.prepare.SampleSeriesWidget.F.	
class method), 567	class_traits()(AFL.automation.prepare.DeckBuilderWidget.VBox
class_trait_names()	class method), 450
	Bakass_traits() (AFL.automation.prepare.PrepareWidget.Button
class method), 575	class method), 477
class_trait_names()	class_traits() (AFL.automation.prepare.PrepareWidget.Checkbox
(AFL.automation.prepare.SampleSeriesWidget.In	
class method), 583	class_traits()(AFL.automation.prepare.PrepareWidget.Dropdown
class_trait_names()	class_traits() (AFL.automation.prepare.rreparewtaget.Dropaown class method), 494
	abalass_traits() (AFL.automation.prepare.PrepareWidget.HBox
class method), 591	class method), 502
class_trait_names()	
(AFL.automation.prepare.SampleSeriesWidget.La	class_traits() (AFL.automation.prepare.PrepareWidget.Label ayout class method), 509
class method), 601	class_traits()(AFL.automation.prepare.PrepareWidget.Layout
class_trait_names()	class_trafts() (AFL.automation.prepare.Freparewtaget.Layout class method), 519
	extlass_traits() (AFL.automation.prepare.PrepareWidget.Text
class method), 611	class method), 532
class_trait_names()	class_traits() (AFL.automation.prepare.PrepareWidget.VBox
(AFL.automation.prepare.SampleSeriesWidget.V.	
class method), 619	class_traits() (AFL.automation.prepare.SampleSeriesWidget.Button
class_trait_names()	class method), 549
	Cutlouss_traits()(AFL.automation.prepare.SampleSeriesWidget.Checkbo
class method), 631	class method), 557
class_trait_names()	class_traits() (AFL.automation.prepare.SampleSeriesWidget.FloatTe.
(AFL.automation.prepare.SweepBuilderWidget.C	
class method), 639	class_traits() (AFL.automation.prepare.SampleSeriesWidget.HBox
class_trait_names()	class method), 575
(Ar L.automation.prepare.SweepВинаerWidget.H	<pre>### IRCD ass_traits() (AFL.automation.prepare.SampleSeriesWidget.IntText</pre>

class method), 583	method), 465
class_traits() (AFL.automation.prepare.SampleSeriesWidleard_c	
class method), 591	method), 468
class_traits() (AFL.automation.prepare.SampleSeriesWidleard_c	nicutory() (AFL.automation.prepare.SampleSeriesWidget.Clier
class method), 601	method), 564
class_traits()(AFL.automation.prepare.SampleSeriesWidlgeatTe	
class method), 611	method), 698
class_traits()(AFL.automation.prepare.SampleSeriesWillgetrV	
class method), 619	method), 704
class_traits()(AFL.automation.prepare.SweepBuilderWillgarB	
class method), 632	AFL.automation.shared.widgetui), 784
class_traits()(AFL.automation.prepare.SweepBuilderWillgarC	<u> </u>
class method), 639	method), 17, 93, 150
class_traits()(AFL.automation.prepare.SweepBuilderWillgarH	
class method), 647	method), 18, 153, 156
class_traits()(AFL.automation.prepare.SweepBuilderWillgarL	
class method), 655	method), 235
class_traits() (AFL.automation.prepare.SweepBuilderWillgarL	
class method), 665	method), 403
class_traits() (AFL.automation.prepare.SweepBuilderWildgarTe	
class method), 674	method), 465
class_traits() (AFL.automation.prepare.SweepBuilderWillgatY	
class method), 682	method), 468
clear() (AFL.automation.APIServer.Driver.PersistentConfglear_	
method), 161	method), 564
clear() (AFL.automation.APIServer.QueueDaemon.DataTeldarun	
method), 179	method), 698
clear() (AFL.automation.loading.OneSelectorBlowoutSample&rll	
method), 268	method), 704
clear() (AFL.automation.loading.PneumaticPressureSampdkCell(e	
method), 278	tribute), 123
clear() (AFL.automation.loading.PneumaticSampleCell.destructed)	
method), 288	method), 387
clear() (AFL.automation.loading.PushPullSelectorSampleCleikkle)	
method), 303	method), 477
method), 303 clear() (AFL.automation.loading.RSoXSSolutionSampleCellide)fc()	
method), 314	method), 548
clear() (AFL.automation.loading.TwoSelectorBlowoutSamplaCkl)	
method), 359	method), 631
clear() (AFL.automation.shared.DatasetWidget.defaultdic€lient	
method), 729	151, 155
clear() (AFL.automation.shared.PersistentConfig.MutableMapprin	
method), 744	24, 791, 792
clear() (AFL.automation.shared.PersistentConfig.PersistentConfig.	
method), 747	233
clear_history() (AFL.automation.APIServer.APIS	
method), 17, 93, 150	401
clear_history()(AFL.automation.APIServer.Client.Cliefdlient	(class in AFL.automation.prepare.OT2Client),

clear_history() (AFL.automation.prepare.DeckBuilderWillenCliebats in AFL.automation.sample.CastingServer),

clear_history() (AFL.automation.prepare.OT2Client.Cliehtent_construct_ui()

clear_history() (AFL.automation.loading.LoadStopperInitionClientSt in AFL.automation.prepare.SampleSeriesWidget),

562

(in

module

method), 18, 153, 156

method), 235

method), 403

- AFL.automation.shared.widgetui), 82, 784, close() (AFL.automation.prepare.SampleSeriesWidget.Text method), 611
- close() (AFL.automation.APIServer.APIServer.FileHandlælose() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 96 method), 619
- close() (AFL.automation.APIServer.APIServer.SMTPHandlerose() (AFL.automation.prepare.SweepBuilderWidget.Button method), 138 method), 632
- close() (AFL.automation.APIServer.APIServer.Zeroconf close() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 149 method), 639
- close() (AFL.automation.prepare.DeckBuilderWidget.Buttohose() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 388 method), 647
- close() (AFL.automation.prepare.DeckBuilderWidget.Che**cklusse**() (AFL.automation.prepare.SweepBuilderWidget.Label method), 396 method), 655
- close() (AFL.automation.prepare.DeckBuilderWidget.Dropdlose() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 409 method), 665
- close() (AFL.automation.prepare.DeckBuilderWidget.HBalose() (AFL.automation.prepare.SweepBuilderWidget.Text method), 417 method), 674
- close() (AFL.automation.prepare.DeckBuilderWidget.Labellose() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 424 method), 682
- close() (AFL.automation.prepare.DeckBuilderWidget.Lay@ltose() (AFL.automation.shared.ServerDiscovery.Zeroconf method), 434 method), 778
- close() (AFL.automation.prepare.DeckBuilderWidget.Textclose_all() (AFL.automation.prepare.DeckBuilderWidget.Button method), 442 class method), 388
- close() (AFL.automation.prepare.DeckBuilderWidget.VBoxlose_all() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 450 class method), 396
- close() (AFL.automation.prepare.PrepareWidget.Button close_all() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 477 class method), 409
- close() (AFL.automation.prepare.PrepareWidget.Checkboxclose_all() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 485 class method), 417
- close() (AFL.automation.prepare.PrepareWidget.Dropdowolose_all() (AFL.automation.prepare.DeckBuilderWidget.Label method), 494 class method), 424
- close() (AFL.automation.prepare.PrepareWidget.HBox close_all() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 502 class method), 434
- close() (AFL.automation.prepare.PrepareWidget.Label close_all() (AFL.automation.prepare.DeckBuilderWidget.Text method), 509 class method), 442
- close() (AFL.automation.prepare.PrepareWidget.Layout close_all() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 519 class method), 450
- close() (AFL.automation.prepare.PrepareWidget.Text close_all() (AFL.automation.prepare.PrepareWidget.Button method), 532 class method), 477
- close() (AFL.automation.prepare.PrepareWidget.VBox close_all() (AFL.automation.prepare.PrepareWidget.Checkbox method), 540 class method), 485
- close() (AFL.automation.prepare.SampleSeriesWidget.Butchose_all() (AFL.automation.prepare.PrepareWidget.Dropdown method), 549 class method), 494
- close() (AFL.automation.prepare.SampleSeriesWidget.Cheddose_all() (AFL.automation.prepare.PrepareWidget.HBox method), 557 class method), 502
- close() (AFL.automation.prepare.SampleSeriesWidget.FloalEse_all() (AFL.automation.prepare.PrepareWidget.Label method), 567 class method), 509
- close() (AFL.automation.prepare.SampleSeriesWidget.HBolose_all() (AFL.automation.prepare.PrepareWidget.Layout method), 575 class method), 519
- close() (AFL.automation.prepare.SampleSeriesWidget.IntEdwse_all() (AFL.automation.prepare.PrepareWidget.Text method), 583 class method), 532
- close() (AFL.automation.prepare.SampleSeriesWidget.Labelose_all() (AFL.automation.prepare.PrepareWidget.VBox method), 591 class method), 540
- close() (AFL.automation.prepare.SampleSeriesWidget.Layakase_all() (AFL.automation.prepare.SampleSeriesWidget.Button method), 601 class method), 549

- close_all() (AFL.automation.prepare.SampleSeriesWidgetchte(AFLxautomation.prepare.DeckBuilderWidget.HBox class method), 557 attribute), 417
- close_all() (AFL.automation.prepare.SampleSeriesWidgeroRho(ATEktautomation.prepare.DeckBuilderWidget.Label class method), 567 attribute), 424
- close_all() (AFL.automation.prepare.SampleSeriesWidgetoHR (AFL.automation.prepare.DeckBuilderWidget.Layout class method), 575 attribute), 435
- close_all() (AFL.automation.prepare.SampleSeriesWidgetchmtTe(AFL.automation.prepare.DeckBuilderWidget.Text class method), 583 attribute), 442
- close_all() (AFL.automation.prepare.SampleSeriesWidgetcharb(AFL.automation.prepare.DeckBuilderWidget.VBox class method), 591 attribute), 450
- close_all() (AFL.automation.prepare.SampleSeriesWidgetchmy(AuFL.automation.prepare.PrepareWidget.Button atclass method), 601 tribute), 477
- close_all() (AFL.automation.prepare.SampleSeriesWidgetoFam(AFL.automation.prepare.PrepareWidget.Checkbox class method), 611 attribute), 485
- close_all() (AFL.automation.prepare.SampleSeriesWidgetoMBAFL.automation.prepare.PrepareWidget.Dropdown class method), 619 attribute), 494
- close_all() (AFL.automation.prepare.SweepBuilderWidgetblatt(AFL.automation.prepare.PrepareWidget.HBox atclass method), 632 tribute), 502
- close_all() (AFL.automation.prepare.SweepBuilderWidgetMihe(AFbxautomation.prepare.PrepareWidget.Label atclass method), 639 tribute), 509
- close_all() (AFL.automation.prepare.SweepBuilderWidgetMBox (AFL.automation.prepare.PrepareWidget.Layout class method), 647 attribute), 520
- close_all() (AFL.automation.prepare.SweepBuilderWidgetoImmbelAFL.automation.prepare.PrepareWidget.Text atclass method), 655 tribute), 532
- close_all() (AFL.automation.prepare.SweepBuilderWidgethmy&AFL.automation.prepare.PrepareWidget.VBox atclass method), 665 tribute), 540
- close_all() (AFL.automation.prepare.SweepBuilderWidget) (AFL.automation.prepare.SampleSeriesWidget.Button class method), 674 attribute), 549
- close_all() (AFL.automation.prepare.SweepBuilderWidgetMR(AFL.automation.prepare.SampleSeriesWidget.Checkbox class method), 682 attribute), 557
- closeConnection() (AFL.automation.loading.ChemyxSyrioganP(AhrjLGheonyaCionspretiane.SampleSeriesWidget.FloatText method), 33, 218, 222 attribute), 568
- collect() (AFL.automation.instrument.FileCamera.FileCam
- collect() (AFL.automation.instrument.NetworkCamera.NetworkCEffnentomation.prepare.SampleSeriesWidget.IntText method), 27, 194 attribute), 583
- collect() (AFL.automation.instrument.SeabreezeUVVis.SeabreeZeUVis.SeabreeZeUVVis.SeabreeZeUVVis.SeabreeZeUVVis.
- collectContinuous() comm (AFL.automation.prepare.SampleSeriesWidget.Layout (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVistribute), 601 method), 28, 205, 207 comm (AFL.automation.prepare.SampleSeriesWidget.Text
- collectSingleSpectrum() attribute), 611
 (AFL.automation.instrument.SeabreezeUVVis.Seabonne(NVV)isutomation.prepare.SampleSeriesWidget.VBox method), 28, 205, 207 attribute), 619
- combine_vars() (AFL.automation.shared.DatasetWidget.Domn&tWillgentomation.prepare.SweepBuilderWidget.Button method), 73, 725, 731 attribute), 632
- combine_vars() (AFL.automation.shared.DatasetWidget.**Rotun**(AWillgattoMudidn.prepare.SweepBuilderWidget.Checkbox method), 73, 727, 731 attribute), 639
- comm (AFL.automation.prepare.DeckBuilderWidget.Button comm (AFL.automation.prepare.SweepBuilderWidget.HBox attribute), 388 attribute), 647
- comm (AFL.automation.prepare.DeckBuilderWidget.Checkb@omm (AFL.automation.prepare.SweepBuilderWidget.Label attribute), 396 attribute), 655
- comm (AFL.automation.prepare.DeckBuilderWidget.Dropdowomm (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 409 attribute), 665

```
comm (AFL.automation.prepare.SweepBuilderWidget.Text copy() (AFL.automation.loading.PneumaticSampleCell.defaultdict
             attribute), 674
                                                                                               method), 288
comm (AFL.automation.prepare.SweepBuilderWidget.VBox copy() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdict
             attribute), 682
                                                                                               method), 303
Component (class in AFL.automation.prepare.Component), copy() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict
             59, 381, 383
                                                                                               method), 314
ComponentDB (class in AFL.automation.prepare), 368
                                                                                  copy() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultd
composition_click_callback()
                                                                                                method), 359
             (AFL.automation.shared.DataLabelerWidget.Datadappe@rWillheatutomation.prepare.Component.Component
             method), 71, 714, 721
                                                                                               method), 59, 381, 384
composition_click_callback()
                                                                                  copy()
                                                                                                       (AFL.automation.prepare.factory.Solution
             (AFL.automation.shared.DatasetWidget.DatasetWidget
                                                                                               method), 690
             method), 73, 725, 731
                                                                                 copy() (AFL.automation.prepare.MassBalance method),
compositionSweepFactory()
                                                                                                371
                                                      (in
                                                                    module
             AFL.automation.prepare), 367
                                                                                  copy() (AFL.automation.prepare.Solute method), 375
compositionSweepFactory()
                                                                    module
                                                                                 copy() (AFL.automation.prepare.Solution method), 377
             AFL.automation.prepare.factory),
                                                                                 copy() (AFL.automation.prepare.Solvent method), 379
                                                                69,
                                                                        688,
                                                                                  copy() (AFL.automation.shared.DatasetWidget.defaultdict
compute_checksum() (AFL.automation.loading.UltimusVPressureController, UltimusVPressureController
                                                                                  CORS (class in AFL.automation.APIServer,APIServer), 93
             method), 56, 363, 364
concentration(AFL.automation.prepare.factory.Solutioncreate_access_token()
                                                                                                                                     (in
                                                                                                                                                      module
             property), 691
                                                                                               AFL.automation.APIServer.APIServer), 84
                               (AFL.automation.prepare.Solution create_expanded_button()
concentration
                                                                                               (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidget
             property), 378
config (AFL.automation.APIServer.APIServer.Flask at-
                                                                                               method), 60, 405, 455
             tribute), 106
                                                                                 create_global_jinja_loader()
config\_class(AFL.automation.APIServer.APIServer.Flask)
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             attribute), 104
                                                                                               method), 108
constrain_samples_conc()
                                                                                  create_jinja_environment()
             (AFL.automation.prepare.MassBalance
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 372
                                                                                               method), 108
                                                                                 \verb|create_queue()| (AFL. automation. APIS erver. APIS
contains() (AFL.automation.prepare.factory.Solution
             method), 690
                                                                                               method), 16, 91, 149
contains()
                                                                                 create_refresh_token()
                                                                                                                                                      module
                               (AFL.automation.prepare.Solution
                                                                                                                                      (in
             method), 377
                                                                                               AFL.automation.APIServer.APIServer), 85
context_processor()
                                                                                  create_url_adapter()
             (AFL.automation.APIServer.APIServer.Flask
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 119
                                                                                               method), 118
continuous_update (AFL.automation.prepare.DeckBuildean WealgeLock() (AFL.automation.APIServer.APIServer.FileHandler
             attribute), 441
                                                                                               method), 96
continuous_update(AFL.automation.prepare.PrepareWidgeaffeet.ock()(AFL.automation.APIServer.APIServer.SMTPHandler
             attribute), 531
                                                                                               method), 138
(AFL.automation.prepare.DeckBuilderWidget.Button
             attribute), 567
continuous_update(AFL.automation.prepare.SampleSeriesWidget.prtffreetty), 388
                                                                                  cross_validation_lock
             attribute), 582
continuous_update (AFL.automation.prepare.SampleSeriesWidget.QaFlL.automation.prepare.DeckBuilderWidget.Checkbox
             attribute), 610
                                                                                               property), 396
continuous_update (AFL.automation.prepare.SweepBuilderoVisignaTixdation_lock
                                                                                               (AFL.automation.prepare.DeckBuilderWidget.Dropdown
             attribute), 673
copy() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defprdpticty), 409
                                                                                  cross_validation_lock
             method), 268
copy() (AFL:automation.loading.PneumaticPressureSampleCell.defa(AFL:automation.prepare.DeckBuilderWidget.HBox
             method), 278
                                                                                               property), 417
```

cross_validation_lock	cross_v	validation_lock
(AFL.automation.prepare.DeckBuilderWidget.La	bel	(AFL.automation.prepare.SampleSeriesWidget.Layout
property), 424		property), 601
cross_validation_lock	cross_v	validation_lock
(AFL.automation.prepare.DeckBuilderWidget.La	yout	(AFL.automation.prepare.SampleSeriesWidget.Text
property), 435		property), 611
cross_validation_lock	cross_v	validation_lock
(AFL.automation.prepare.DeckBuilderWidget.Tex	xt	(AFL.automation.prepare.SampleSeriesWidget.VBox
property), 442		property), 619
cross_validation_lock	cross_v	validation_lock
(AFL. automation. prepare. Deck Builder Widget. VB)	<i>3ox</i>	(AFL. automation. prepare. Sweep Builder Widget. Button
property), 450		property), 632
cross_validation_lock	cross_v	validation_lock
(AFL. automation. prepare. Prepare Widget. Button		(AFL.automation.prepare.SweepBuilderWidget.Checkbox
property), 477		property), 640
cross_validation_lock	cross_v	validation_lock
(AFL. automation. prepare. Prepare Widget. Checkbox and the content of the cont	oox	(AFL.automation.prepare.SweepBuilderWidget.HBox
property), 485		property), 647
cross_validation_lock	cross_v	alidation_lock
(AFL. automation. prepare. Prepare Widget. Dropdomation and the properties of the	own	(AFL.automation.prepare.SweepBuilderWidget.Label
property), 494		property), 655
cross_validation_lock	cross_v	alidation_lock
(AFL. automation. prepare. Prepare Widget. HBox		(AFL.automation.prepare.SweepBuilderWidget.Layout
property), 502		property), 665
cross_validation_lock	cross_v	validation_lock
(AFL.automation.prepare.PrepareWidget.Label		(AFL.automation.prepare.SweepBuilderWidget.Text
property), 509		property), 674
cross_validation_lock		validation_lock
(AFL.automation.prepare.PrepareWidget.Layout		(AFL.automation.prepare.SweepBuilderWidget.VBox
property), 520	10 (4	property), 682
cross_validation_lock	cwd()(A	AFL.automation.instrument.SeabreezeUVVis.Path
(AFL.automation.prepare.PrepareWidget.Text		class method), 199
property), 532	D	
cross_validation_lock	_	
(AFL.automation.prepare.PrepareWidget.VBox property), 540	daemon((AFL.automation.APIServer.APIServer.QueueDaemon property), 135
cross_validation_lock	daemon ((AFL.automation.APIServer.QueueDaemon.QueueDaemon
(AFL. automation. prepare. Sample Series Widget. Between the property of the	utton	property), 180
property), 549	daemon ((AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDae
cross_validation_lock		property), 794
(AFL. automation. prepare. Sample Series Widget. Constant and the property of the property o	haakehoon ((AFL. automation. loading. Load Stopper Driver. Sensor Polling Threads and Stopper Driver. Sensor Polling Threads are the state of th
property), 557		property), 243
cross_validation_lock	daemon ((AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
(AFL. automation. prepare. Sample Series Widget. Figure 1.00% and 1.00% and 1.00% are also below to the contract of the cont	loatText	property), 246
property), 568	daemon((AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
cross_validation_lock		property), 249
(AFL. automation. prepare. Sample Series Widget. H	$R_{ m Paremon}$	(AFL.automation.loading.Sensor.DummySensor1
property), 575		property), 322
cross_validation_lock	daemon ((AFL.automation.loading.Sensor.DummySensor2
(AFL. automation. prepare. Sample Series Widget. In the prepare of the prepare	<i>tText</i>	property), 325
property), 583	daemon ((AFL.automation.loading.SensorCallbackThread.SensorCallbackT
cross_validation_lock		property), 331
(AFL. automation. prepare. Sample Series Widget. Loss of the contract of the	a laa lemon ((AFL. automation. loading. Sensor Callback Thread. Simple Thresholds and the sensor of the sensor
property), 591		property), 333

property), 333

	AFL.automation.loading	.SensorCallbackThred				
	property), 336			(AFL. automation. sample	.CastingServer.OT2	Client
daemon (A	AFL. $automation$. $loading$	c. Sensor Callback Three				
	property), 339			ass in AFL.automation.pre	epare), 369	
daemon (A	AFL. $automation$. $loading$	s. Sensor Polling Thread	.Sen DocRoBlin	∦ <i>tilæn</i> twiddget	(class	in
	property), 345			AFL.automation.prepare	.DeckBuilderWidge	t),
daemon (A	AFL. automation. shared.	ServerDiscovery.RunT	Thread	60, 404, 455		
	property), 761		DeckBui	.lderWidget	(class	in
	AFL.automation.shared. property), 767	ServerDiscovery.Servi	iceBrowser	<i>AFL.automation.prepare</i> 490	e.PrepareWidget),	
		neumaticPressureSam	nle Ciella RiBerri	nlidieir Miedig ente Stiondpelle Cell	(class	in
	property), 40, 275, 279		.	AFL.automation.prepare		
	elerModel	(class	in	60, 404, 455		/,
	AFL.automation.shared	*			(class	in
	71, 712, 722	,,		AFL.automation.prepare	,	
	elerView	(class	in	60, 405, 455		,,
	AFL.automation.shared	•		() (AFL.automation.instru	ment SeabreezeUVV	lis Fa
	71, 712, 722		uccouc	class method), 197	mem.seaoreezee v v	15.24
	elerWidget	(class	in decode_	key_loader()		
	AFL.automation.shared			(AFL.automation.APISer	rver.APIServer.JWT	Manager
	70, 713, 721			method), 130		
		d. DatasetWidget. Datas	setW ildpeod/Ad	dphoperties (AFL.auto	mation.APIServer.A	PIServer ServiceInfo
	property), 73, 726, 731	z araser // tageriz aras	, e. , , <u></u>	attribute), 142		. iso, rouse, receny
Dataset		(class	<i>in</i> decoded	l_properties(<i>AFL.auto</i> a	mation shared Serve	rDiscovery AsyncSei
	AFL.automation.shared	•		attribute), 754		. 2 1,500 (0.) 11 15) 11 0 50 1
	72, 722, 730	.Daraser (rager),	decoded	l_properties(<i>AFL.auto</i> a	mation shared Serve	rDiscovery ServiceIi
	Widget_Model	(class	in	attribute), 771	manon.snarea.serve	TDiscover y.serviceir
	AFL.automation.shared	*		_config(AFL.automatio	on APIServer APISer	ver Flask
	73, 726, 731	.Daraser (rager),	uciuui	attribute), 105	11.211 1501 101.211 1501	ver.1 tusk
	Widget_View	(class	in default	_factory(<i>AFL.automati</i>	ion loading OneSele	ctorRlowoutSample(
	AFL.automation.shared		in acraare	attribute), 268	on.ioaaing.oneseie	ciorBiowouisampiec
	74, 727, 732	.Dataset (raget),	default	_factory(<i>AFL.automati</i>	ion loadina Pneuma	ticPressureSampleCa
DataTra		class	in	attribute), 278	on.todding.1 neuma	ner ressuresumple ee
	AFL.automation.APISe			_factory(<i>AFL.automati</i>	ion loadina Pneuma	ticSampleCell defaul
	177	rver.QueueDuemon),	deraure	attribute), 288	on.ioaaing.i neama	исвитри Сен.иејин
deactiv		nation prepare Dumm	v OTAFAII	er . Daonony (AIEL Duiven att	ion loadina PushPul	llSelectorSampleCell
	method), 61, 460, 462	паноп.ргераге.Винип	y_Oue_bume	attribute), 303	on.ioaaing.i usni ui	iseieciorsampieceii
		IServer APIServer Fla	\mathfrak{s}^k default	_factory(<i>AFL.automati</i>	ion loading RSoXSS	olutionSampleCell de
	property), 109		on acraare	attribute), 314	01110000115011501155	suitensample centa
		rver APIServer APISe	rver de fault	_factory(<i>AFL.automati</i>	ion loading TwoSele	ctorRlowoutSample(
ucbug()	method), 17, 93, 150	7 707.211 1507 707.211 150	, ver deradre	attribute), 359	on.todding.1wosete	cioi Biowoui sampie e
debug()		APIServer Client Clie	nt default	_factory(<i>AFL.automati</i>	ion shared DatasetW	Vidaet defaultdict
ucbug()	method), 18, 153, 156	In igerver. enem. ene	ni acraare	attribute), 729	on.snarca.Danasci vi	iagei.aejaiiiaiei
debua()	(AFL.automation.loadin	g.LoadStopperDriver.	<i>Cliedt</i> e fault		'ass	in
ucbug ()	method), 235	.8. Дойивторрет В ттет.	Circulation and the	AFL.automation.loading		
dehua()	(AFL.automation.prepar	re DeckRuilderWidget	Client	267	.oneseteetor Brower	iisampie ceit),
ucbug ()	method), 403	e.BeenBuitaer ii tagen	default		'ass	in
debug()	* *	repare.OT2Client.Clie		AFL.automation.loading		
ucbug()	<i>method</i>), 465	epare.012enem.ene		277	.1 neumanel ressure	sample Cell);
dehua()	(AFL.automation.prepar	re OT2Client OT2Clie	nt default		'ass	in
ucoug ()	method), 468		acraure	AFL.automation.loading		
debua()	(AFL.automation.prepar	re SampleSeriesWidoe	t.Client	287	neumanesampleC	,
ug()	method), 564	. c.sampieseries iriuge	default		'ass	in
debua()	(AFL.automation.sam)	ole CastingServer Clie		AFL.automation.loading		
5 ()						r · · · · · · · · / / /

```
302
                                                                                                                                                                           method), 165
defaultdict
                                                                                                                                        in deposit_obj() (AFL.automation.APIServer.DummyDriver.DummyDriver
                                                                               (class
                        AFL.automation.loading.RSoXSSolutionSampleCell),
                                                                                                                                                                           method), 168
                                                                                                                                                   deposit_obj() (AFL.automation.APIServer.DummyOT2Driver.Driver
defaultdict
                                                                               (class
                                                                                                                                                                            method), 172
                        AFL.automation.loading.TwoSelectorBlowoutSamplepOstit_obj() (AFL.automation.APIServer.DummyOT2Driver.DummyDt
                        358
                                                                                                                                                                            method), 174
defaultdict
                                                                                                                                        in deposit_obj() (AFL.automation.instrument.DummySAS.Driver
                        AFL.automation.shared.DatasetWidget),
                                                                                                                                                                            method), 185
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.DummySAS.DummySAS
defaults (AFL.automation.APIServer.DummyDriver.DummyDriver method), 187
                        attribute), 21, 168, 169
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.I22SAXS.Driver
defaults (AFL.automation.APIServer.DummyOT2Driver.DummyDrimethod), 190
                        attribute), 21, 174, 175
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.I22SAXS.I22SAXS
{\tt defaults} \ (AFL. automation. instrument. Dummy SAS. Dummy SAS
                                                                                                                                                                            method), 192
                        attribute), 26, 186, 188
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.SeabreezeUVVis.Driver
defaults (AFL.automation.instrument.I22SAXS.I22SAXS
                                                                                                                                                                            method), 196
                        attribute), 26, 192, 193
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeU
defaults (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVWithod), 205
                        attribute), 27, 205, 206
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.SpecScreen_Driver.Driver
defaults (AFL.automation.loading.LoadStopperDriver.LoadStopperDreithard), 210
                        attribute), 36, 240, 250
                                                                                                                                                   deposit_obj() (AFL.automation.instrument.SpecScreen_Driver.SpecScre
{\tt defaults} (AFL. automation. loading. One Selector Blowout Sample Cell. \textit{One the Sedde} control of the Cell o
                                                                                                                                                   deposit_obj() (AFL.automation.loading.LoadStopperDriver.Client
                        attribute), 261
defaults (AFL.automation.loading.OneSelectorBlowoutSampleCell. TreedSedles CO3BlowoutSampleCell
                        attribute), 265
                                                                                                                                                   deposit_obj() (AFL.automation.loading.LoadStopperDriver.Driver
defaults (AFL.automation.loading.PneumaticPressureSampleCell.Pmeuhocat); PressureSampleCell
                                                                                                                                                   deposit\_obj() (AFL.automation.loading.LoadStopperDriver.LoadStoppe
                        attribute), 39, 274, 279
defaults (AFL.automation.loading.PneumaticSampleCell.PneumatioSathplieCellO
                        attribute), 41, 284, 288
                                                                                                                                                   deposit_obj() (AFL.automation.loading.OneSelectorBlowoutSampleCell
defaults (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolutionSampleCell
                        attribute), 45, 310, 315
                                                                                                                                                   deposit\_obj() (AFL. automation. loading. One Selector Blowout Sample Cells deposit\_obj() and the selection of the selectio
defaults (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwebSolex; (2018) to Manager and the faults (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwebSolex; (2018) to Manager and Manage
                                                                                                                                                   deposit\_obj() (AFL.automation.loading.OneSelectorBlowoutSampleCelling)
                        attribute), 55, 356, 360
defaults (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OTiethDrd)\ellow266
                        attribute), 61, 459, 461
                                                                                                                                                   deposit_obj() (AFL.automation.loading.PneumaticPressureSampleCell...
defaults (AFL.automation.sample.CastingServer.CastingServer
                                                                                                                                                                           method), 272
                        attribute), 82, 695, 705
                                                                                                                                                   deposit_obj() (AFL.automation.loading.PneumaticPressureSampleCell...
defaults (AFL.automation.sample_env.TemperatureDeck.TemperatumeDeckl), 275
                        attribute), 69, 709, 710
                                                                                                                                                   deposit_obj() (AFL.automation.loading.PneumaticSampleCell.Driver
delete() (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                            method), 282
                        method), 119
                                                                                                                                                   deposit_obj() (AFL.automation.loading.PneumaticSampleCell.Pneumat
density (AFL.automation.prepare.Component.Component
                                                                                                                                                                           method), 285
                        property), 59, 382, 384
                                                                                                                                                   deposit\_obj() (AFL.automation.loading.PushPullSelectorSampleCell.Displayers)
density (AFL.automation.prepare.Solute property), 375
                                                                                                                                                                            method), 296
```

deposit_obj() (AFL.automation.APIServer.Driver.Driverdeposit_obj() (AFL.automation.loading.TwoSelectorBlowoutSampleCellimethod), 20, 159, 163 method), 352
deposit_obj() (AFL.automation.APIServer.DummyDriverdDpisit_obj() (AFL.automation.loading.TwoSelectorBlowoutSampleCellimethod)

deposit_obj() (AFL.automation.APIServer.APISer

deposit_obj() (AFL.automation.APIServer.Client.Client deposit_obj() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX

method), 300

method), 307

method), 311

deposit_obj() (AFL.automation.loading.PushPullSelectorSampleCell.Pu

Index 825

density (AFL.automation.prepare.Solvent property),

method), 17, 92, 150

method), 18, 153, 156

method), 357	attribute), 591
deposit_obj() (AFL.automation.prepare.DeckBuilderWidgesChirp	**
method), 403	attribute), 611
deposit_obj() (AFL.automation.prepare.Dummy_OT2_DdiescrIm)	
method), 457	attribute), 630
deposit_obj() (AFL.automation.prepare.Dummy_OT2_DatescrExp	
	attribute), 640
method), 460	
deposit_obj() (AFL.automation.prepare.OT2Client.Cliendescrip	
method), 465	attribute), 655
deposit_obj() (AFL.automation.prepare.OT2Client.OT2Clien	
method), 468	attribute), 674
deposit_obj() (AFL.automation.prepare.SampleSeriesWidesCollip	
method), 564	(AFL.automation.prepare.DeckBuilderWidget.Checkbox
${\tt deposit_obj()} \ (AFL. automation. sample. Casting Server. Casting Server$	
	tion_allow_html
deposit_obj()(AFL.automation.sample.CastingServer.Client	(AFL.automation.prepare.DeckBuilderWidget.Dropdown
method), 699	attribute), 409
${\tt deposit_obj()} \ (AFL. automation. sample. Casting Server. D{\tt rdes} {\tt crip}$	tion_allow_html
method), 701	(AFL.automation.prepare.DeckBuilderWidget.Label
${\tt deposit_obj()} \ (AFL. automation. sample. Casting Server. OT 2 Client$	attribute), 425
	tion_allow_html
deposit_obj() (AFL.automation.sample_env.TemperatureDeck.Dru	iverFL.automation.prepare.DeckBuilderWidget.Text
method), 707	attribute), 442
deposit_obj() (AFL.automation.sample_env.Temperaturedlesskrffep	
method), 709	(AFL.automation.prepare.PrepareWidget.Checkbox
description (AFL.automation.prepare.DeckBuilderWidget.Button	
- · · · · · · · · · · · · · · · · · · ·	tion_allow_html
description (AFL.automation.prepare.DeckBuilderWidget.Checkbo	
attribute), 396	attribute), 494
description (AFL.automation.prepare.DeckBuilderWidgedDsapdp	
	(AFL.automation.prepare.PrepareWidget.Label
attribute), 409	
description(AFL.automation.prepare.DeckBuilderWidget.Label	attribute), 510
	tion_allow_html
description(AFL.automation.prepare.DeckBuilderWidget.Text	(AFL.automation.prepare.PrepareWidget.Text
attribute), 442	attribute), 532
${\tt description} (AFL. automation. prepare. Prepare Widget. But {\tt des} {\tt crip})$	
attribute), 476	(AFL. automation. prepare. Sample Series Widget. Checkbox
${\tt description} (AFL. automation. prepare. Prepare Widget. Checkbox$	attribute), 557
	tion_allow_html
${\tt description} (AFL. automation. prepare. Prepare Widget. Drop down$	(AFL. automation. prepare. Sample Series Widget. Float Text
attribute), 494	attribute), 568
${\tt description}$ $(AFL. automation. prepare. Prepare Widget. Labelescription)$	tion_allow_html
attribute), 509	(AFL.automation.prepare.SampleSeriesWidget.IntText
description(AFL.automation.prepare.PrepareWidget.Text	attribute), 583
	tion_allow_html
${\tt description} (AFL. automation. prepare. Sample Series Widget. Button$	
attribute), 548	attribute), 591
description (AFL.automation.prepare.SampleSeriesWidgeteSharikh	
attribute), 557	(AFL.automation.prepare.SampleSeriesWidget.Text
description (AFL.automation.prepare.SampleSeriesWidget.FloatTe	
	tion_allow_html
description (AFL.automation.prepare.SampleSeriesWidget.IntText	
attribute), 583	attribute), 640
description (AFL.automation.prepare.SampleSeriesWidgetelsadnilp	12

	n.prepare.SweepBuilder	Widget.Label	AFL.automation.shared.	serialization),	80,
attribute), 655	-	-1.55	781, 782	. 1	
description_allow_htm			ctionLabeler	(class	in
(AFL.automation attribute), 675	n.prepare.SweepBuilder	Widget.Text	AFL.automation.shared. 74, 733, 741	DiffractionLabele	<i>r</i>),
description_tooltip		Diffrac	ctionLabelerModel	(class	in
	n.prepare.DeckBuilderW	idget.Checkbox	AFL.automation.shared.	DiffractionLabele	(r),
property), 396	1 1	O	75, 734, 741	35	,,
description_tooltip		Diffrac	ctionLabelerView	(class	in
	n.prepare.DeckBuilderW		AFL.automation.shared.	DiffractionLabele	(r),
property), 409	1 1		75, 734, 741	30	,,
description_tooltip		Digital	OutPressureControll	er (class	in
	n.prepare.DeckBuilderW	_	AFL.automation.loading		ureController),
property), 425			33, 223, 225		,,
description_tooltip		disable	ed (AFL.automation.prepa	re.DeckBuilderW	idget.Button
	n.prepare.DeckBuilderW		attribute), 387		
property), 442	Transaction of the second of t		ed (AFL.automation.prepa	re.DeckBuilderW	idget.Checkbox
description_tooltip			attribute), 396		
	n.prepare.PrepareWidge	et.Checkbodi sable	ed (AFL.automation.prepa	re DeckBuilderW	idget Drondown
property), 485	inpropulser repuise (ridge		attribute), 409		a.gen.z ropuo mi
description_tooltip		disable	ed (AFL.automation.prepa	re DeckRuilderW	idøet Text
	n.prepare.PrepareWidge		attribute), 441	re.BeenBinaer III	ageem
property), 494	i.prepare.i repare mase		ed (AFL.automation.prepa	re PrenareWidoet	Rutton
description_tooltip		disasi	attribute), 476	re.i repare magei	.Button
	n nrenare PrenareWidae	ot Label disable	ed (AFL.automation.prepa	re PrenareWidget	Checkhor
property), 510	i.prepare.i repare mage	i.Lubei disabi	attribute), 485	re.1 repare magei	.Checkbox
description_tooltip		disahla	ed (AFL.automation.prepa	re PrenareWidget	Drondown
	n.prepare.PrepareWidge		attribute), 494	re.i repare magei	.Вториотп
property), 532	i.prepare.i repare mage		ed (AFL.automation.prepa	are PrepareWidge	t Text
description_tooltip		ursabre	attribute), 531	ire.1 repare wage	ι. Ι Ελί
	n nranara SamplaSarias	Widget Chairlebah	ed (AFL.automation.prepa	ura SamplaSarias V	Vidaet Rutton
property), 557	i.prepare.sumpieseries	mager.Chausable	attribute), 548	re.sumpleseries vi	ragei.Buiton
description_tooltip		dicable	ed (AFL.automation.prepa	ura SamplaSarias V	Vidaat Chackbox
	n.prepare.SampleSeries\			re.sumpleseries vi	ragei.Checkbox
property), 568	i.prepare.sampieseries		ed (AFL.automation.prepa	ura CamplaCariasU	Vidaat FloatTaxt
description_tooltip		uisabie	attribute), 566	re.sampieseries w	riagei.FioaiTexi
	n nuanana Campla Cariasi	Widest Intifficated	ed (AFL.automation.prepa	una CamplaCariasU	Vidaat IntTaxt
	i.prepare.sampieseries		attribute), 582	re.sampieseries w	riagei.IniTexi
property), 583				una CamplaCariasU	Widget Toxt
description_tooltip	n nuanana CamplaCaniasi		ed (AFL.automation.prepa	re.sampieseriesw	riagei.Texi
	n.prepare.SampleSeries		attribute), 610	una Cunaan Duildan I	Widget Dutton
<pre>property), 591 description_tooltip</pre>		uisabie	ed (AFL.automation.prepa	re.sweeр ь инаеrv	viagei. Б иноп
		(V: J T	attribute), 631	C D:11	W. J. of Cl. of l.l.
	i.prepare.sampieseries	wiagei. Tex u isabie	ed (AFL.automation.prepa	re.sweeр ь инаеrv	viagei.Cneckbox
property), 611		ا المام	attribute), 640	C D:11	W: 14 T4
description_tooltip	C D :1.1		ed (AFL.automation.prepa	re.Sweepвинаerv	viaget.1ext
	n.prepare.SweepBuilder	~			
property), 640		aiscove	er_server_by_name()	CI: . C	D.
description_tooltip	C D 11	TIP 1 1 1 1	(AFL.automation.APISer	rver.Ciient.Server	Discovery
	n.prepare.SweepBuilder		method), 154		
property), 655		alscove	er_server_by_name()	IC D:	C D:
description_tooltip	n man and Court D '11	Wident Tour	(AFL.automation.shared	.serverDiscovery.	ServerDiscovery
	n.prepare.SweepBuilder		method), 79, 764, 779	A DIC	ADIC El 1
property), 675	()		ch_request() (AFL.auto	mation.APIServer	.APIServer.Flask
deserialize()	(in	module	method), 116		

```
dispense() (AFL.automation.loading.ChemyxSyringePumph(SheproixShein(QePAIFI)pautomation.APIServer.APIServer.ServiceInfo
             method), 32, 220, 222
                                                                                               method), 142
dispense() (AFL.automation.loading.ChemyxSyringePumptsyringerPumpt() (AFL.automation.shared.ServerDiscovery.AsyncServiceIn
                                                                                               method), 754
             method), 221
dispense() (AFL.automation.loading.DummyPump.Dumnddsuppointer() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 35, 230, 231
                                                                                               method), 771
dispense() (AFL.automation.loading.DummyPump.SyringhtSuppervice() (AFL.automation.APIServer.APIServer.ServiceInfo
             method), 231
                                                                                               method), 143
dispense() (AFL.automation.loading.NE1kSyringePump.MEkSpringePump.AFL.automation.shared.ServerDiscovery.AsyncServiceIn
             method), 38, 253, 255
                                                                                               method), 755
dispense() (AFL.automation.loading.NE1kSyringePump.Syminge() (AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                               method), 772
             method), 255
dispense() (AFL:automation.loading.PressureControllerAdrsungeRudeLoatstoHuntschlungeResearcherter.APIServer.ServiceInfo
             method), 43, 292, 294
                                                                                               method), 143
dispense() (AFL.automation.loading.PressureControllerAcksurteSyringAlFilmputomation.shared.ServerDiscovery.AsyncServiceInfo
             method), 293
                                                                                               method), 755
dispense() (AFL.automation.loading.SyringePump.SyringeRsumext() (AFL.automation.shared.ServerDiscovery.ServiceInfo
             method), 54, 348
                                                                                               method), 772
dispense_rate() (AFL.automation.prepare.OT2Client.OTBC/lientrdown_appcontext()
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 62, 468, 470
dispense_rate() (AFL.automation.sample.CastingServer.OT2Clienmethod), 125
             method), 703
                                                                                 do_teardown_request()
dispenseRunning() (AFL.automation.loading.DigitalOutPressureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHeessureConfidentDigitalOutHees
             method), 224
                                                                                               method), 125
dispenseRunning() (AFL.automation.loading.DigitalOutRoassu(ArClantrolbeautronsshureEdoSterokleDiscovery.AsyncServiceBrowser
             method), 225
                                                                                               attribute), 751
dispenseRunning() (AFL.automation.loading.PressureColored & HELasstone & Server Discovery.Service Browser
             method), 42, 290
                                                                                               attribute), 766
dispenseRunning() (AFL.automation.loading.UltimusVPDoxiblesVairtiMidrtEposSeeXGoottoaller
                                                                                                                                           (class
             method), 362
                                                                                               AFL.automation.loading.DoubleViciMultiposSelector),
dispenseRunning() (AFL.automation.loading.UltimusVPressureController
             method), 363
                                                                                 draw_deck() (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderW
display (AFL.automation.prepare.DeckBuilderWidget.Layout
                                                                                               method), 60, 405, 455
                                                                                 draw_peaks() (AFL.automation.shared.DataLabelerWidget.DataLabelerV
             attribute), 432
display (AFL.automation.prepare.PrepareWidget.Layout
                                                                                               method), 71, 714, 721
                                                                                 draw_peaks() (AFL.automation.shared.DiffractionLabeler.DiffractionLab
             attribute), 517
display (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                                                               method), 75, 733, 741
             attribute), 599
                                                                                 {\tt drive}\, (AFL. automation. instrument. Seabreeze UVV is. Path
display (AFL.automation.prepare.SweepBuilderWidget.Layout
                                                                                               property), 202
             attribute), 663
                                                                                 Driver (class in AFL.automation.APIServer.Driver), 19,
display()
                                                                    module
             AFL.automation.shared.widgetui), 784
                                                                                 Driver (class in AFL.automation.APIServer.DummyDriver),
dns_addresses() (AFL.automation.APIServer.APIServer.ServiceInf@64
                                                                                 Driver (class in AFL.automation.APIServer.DummyOT2Driver),
             method), 142
dns_addresses() (AFL.automation.shared.ServerDiscovery.AsyncSarMiceInfo
                                                                                 Driver (class in AFL.automation.instrument.DummySAS),
             method), 754
dns_addresses() (AFL.automation.shared.ServerDiscovery.ServiceInflo
                                                                                 Driver (class in AFL.automation.instrument.I22SAXS),
             method), 771
dns_nsec() (AFL.automation.APIServer.APIServer.ServiceInfo
                                                                                               189
             method), 142
                                                                                 Driver (class in AFL.automation.instrument.SeabreezeUVVis),
dns_nsec() (AFL.automation.shared.ServerDiscovery.AsyncServiceInflet
                                                                                 Driver (class in AFL.automation.instrument.SpecScreen_Driver),
             method), 754
dns_nsec() (AFL.automation.shared.ServerDiscovery.ServiceInfo 208
             method), 771
                                                                                 Driver (class in AFL. automation. loading. Load Stopper Driver),
```

```
235
                                                                                   dummy_reset_tank_levels()
Driver (class in AFL.automation.loading.OneSelectorBlowoutSample(AHII), automation.APIServer.DummyDriver.DummyDriver
                                                                                                 method), 21, 168, 170
Driver (class in AFL.automation.loading.PneumaticPressuckSimmyplrestlt,_tank_levels()
                                                                                                 (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
Driver (class in AFL.automation.loading.PneumaticSampleCell),
                                                                                                 method), 22, 174, 176
                                                                                   DummvDriver
                                                                                                                                (class
                                                                                                                                                                 in
Driver (class in AFL.automation.loading.PushPullSelectorSampleCelA)FL.automation.APIServer.DummyDriver),
                                                                                                 21, 166, 169
Driver (class in AFL.automation.loading.RSoXSSolutionSaDunderGebriver
                                                                                                                                (class
                                                                                                 AFL.automation.APIServer.DummyOT2Driver),
Driver (class in AFL.automation.loading.TwoSelectorBlowoutSample@elfl\)72, 175
                                                                                   DummyPump (class in AFL.automation.loading.DummyPump),
Driver (class in AFL.automation.prepare.Dummy_OT2_Driver),
                                                                                                 35, 229, 231
                                                                                   DummySAS (class in AFL.automation.instrument.DummySAS),
Driver (class in AFL. automation. sample. Casting Server),
                                                                                                 25, 185, 188
                                                                                   DummySensor1
                                                                                                                                 (class
                                                                                                                                                                 in
Driver (class in AFL. automation. sample env. Temperature Deck),
                                                                                                 AFL.automation.loading.Sensor),
                                                                                                                                                             321,
driver_status() (AFL.automation.APIServer.APIServer.ApithterSensor2
                                                                                                                                 (class
                                                                                                                                                                 in
             method), 16, 92, 150
                                                                                                 AFL.automation.loading.Sensor),
                                                                                                                                                             324.
driver_status() (AFL.automation.APIServer.Client.Client
             method), 18, 153, 155
driver_status() (AFL.automation.loading.LoadStopperDriver.Client
             method), 234
                                                                                   emit() (AFL.automation.APIServer.APIServer.FileHandler
driver_status() (AFL.automation.loading.Sensor.DummySensor2 method), 96
             method), 49, 325, 328
                                                                                   emit() (AFL.automation.APIServer.APIServer.SMTPHandler
driver_status()(AFL.automation.prepare.DeckBuilderWidget.Cliqnethod), 137
             method), 403
                                                                                   emit()(AFL.automation.prepare.Component.Component
driver_status() (AFL.automation.prepare.OT2Client.Client
                                                                                                 method), 59, 381, 383
              method), 464
                                                                                   emit() (AFL.automation.prepare.Solute method), 375
driver_status() (AFL.automation.prepare.OT2Client.OTeMS.hight (AFL.automation.prepare.Solvent method), 379
             method), 468
                                                                                   emit_protocol() (AFL.automation.prepare.OT2Client.PipetteAction
driver_status() (AFL.automation.prepare.SampleSeriesWidget.Climethod), 470
             method), 564
                                                                                   emit_protocol() (AFL.automation.prepare.PipetteAction
driver_status() (AFL.automation.sample.CastingServer.Client
                                                                                                 method), 373
             method), 698
                                                                                   emit_protocol()
                                                                                                                    (AFL. automation. prepare. Sample \\
driver_status() (AFL.automation.sample.CastingServer.OT2Clienmethod), 373
             method), 704
                                                                                   Empty, 743
Dropdown (class in AFL.automation.prepare.DeckBuilderWiehgerty () (AFL.automation.APIServer.APIServer.MutableQueue
                                                                                                 method), 133
Dropdown (class in AFL.automation.prepare.PrepareWidgetempty() (AFL.automation.shared.MutableQueue.MutableQueue
                                                                                                 method), 76, 742, 743
drySyringe() (AFL.automation.loading.OneSelectorBlowomtSuppleCephiOneSele7801BlowoutSampleCell
             method), 39, 261, 269
                                                                                   emptySyringe() (AFL.automation.loading.ChemyxSyringePump.Chemyx.
drySyringe() (AFL.automation.loading.OneSelectorBlowoutSampleGellTwpSelectorBlowoutSampleCell
             method), 266
                                                                                   emptySyringe() (AFL.automation.loading.ChemyxSyringePump.SyringeF
drySyringe() (AFL.automation.loading.RSoXSSolutionSampleCell.R&aK&SqlWionSampleCell
             method), 46, 310, 316
                                                                                   emptySyringe() (AFL.automation.loading.DummyPump.DummyPump
drySyringe() (AFL.automation.loading.TwoSelectorBlowoutSampleGelhTilySeleCtatBlowoutSampleCell
             method), 56, 356, 361
                                                                                   \verb"emptySyringe" () (AFL. automation. loading. DummyPump. SyringePump
Dummy_OT2_Driver
                                                 (class
                                                                             in
                                                                                                 method), 231
             AFL. automation. prepare. Dummy\_OT2\_Driver), \ \ \texttt{emptySyringe}() \ (AFL. automation. loading. NE1kSyringePump. NE1kSyring
             61, 458, 461
                                                                                                 method), 38, 254, 255
```

method), 704

```
emptySyringe() (AFL.automation.loading.NE1kSyringePuenpuStyeijegeRuft)p(AFL.automation.APIServer.APIServer.Flask
             method), 255
                                                                                                method), 116
emptySyringe() (AFL.automation.loading.PressureControllems.flums.pinPressureControllems.prepType), 471
             method), 43, 292, 294
                                                                                  env (AFL.automation.APIServer.APIServer.Flask prop-
emptySyringe() (AFL.automation.loading.PressureControllerAsPunapt,SyringePump
             method), 293
                                                                                  environment variable
emptySyringe() (AFL.automation.loading.SyringePump.SyringePLASKQ_DEBUG, 110
                                                                                  {\tt Eq}\ (class\ in\ AFL. automation. instrument. Seabreeze UVV is),
             method), 54, 348
encode() (AFL.automation.instrument.SeabreezeUVVis.Eq
                                                                                  \verb|error_handler_spec| (AFL. automation. APIS erver. APIS erver. Flask
             method), 197
encode_key_loader()
                                                                                                attribute), 123
             (AFL.automation.APIServer.APIServer.JWTManagerorhandler() (AFL.automation.APIServer.APIServer.Flask
             method), 130
                                                                                                method), 120
endpoint() (AFL.automation.APIServer.APIServer.Flask example_label_cb() (AFL.automation.prepare.PrepareWidget.SampleSe
             method), 119
                                                                                                method), 526
enforce_units()
                                              (in
                                                                     module
                                                                                  example_label_cb() (AFL.automation.prepare.SampleSeriesWidget.Sam
             AFL.automation.prepare.Component), 380
                                                                                                method), 65, 606, 624
enforce_units()
                                                                     module
                                                                                  execute()
                                                                                                       (AFL.automation.APIServer.Driver.Driver
                                              (in
             AFL.automation.shared.units),
                                                                         782.
                                                                                                method), 20, 159, 162
                                                                                  execute() (AFL.automation.APIServer.DummyDriver.Driver
enqueue() (AFL.automation.APIServer.APIServer.APIServer
                                                                                                method), 165
             method), 17, 93, 150
                                                                                  execute() (AFL.automation.APIServer.DummyDriver.DummyDriver
                     (AFL.automation.APIServer.Client.Client
enqueue()
                                                                                                method), 168
             method), 18, 153, 155
                                                                                  execute() (AFL.automation.APIServer.DummyOT2Driver.Driver
enqueue() (AFL.automation.loading.LoadStopperDriver.Client
                                                                                                method), 172
                                                                                  {\tt execute()} \ (AFL. automation. APIS erver. Dummy OT2 Driver. Dummy Driver
             method), 235
enqueue() (AFL.automation.prepare.DeckBuilderWidget.Client
                                                                                                method), 21, 174, 175
                                                                                  execute() (AFL.automation.instrument.DummySAS.Driver
             method), 403
enqueue() (AFL.automation.prepare.OT2Client.Client
                                                                                                method), 185
             method), 465
                                                                                  execute() (AFL.automation.instrument.DummySAS.DummySAS
enqueue() (AFL.automation.prepare.OT2Client.OT2Client
                                                                                                method), 187
             method), 468
                                                                                  execute() (AFL.automation.instrument.I22SAXS.Driver
enqueue() (AFL.automation.prepare.SampleSeriesWidget.Client
                                                                                                method), 190
             method), 564
                                                                                  execute() (AFL.automation.instrument.I22SAXS.I22SAXS
enqueue() (AFL.automation.sample.CastingServer.Client
                                                                                                method), 192
                                                                                  execute() (AFL.automation.instrument.SeabreezeUVVis.Driver
             method), 698
enqueue() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                method), 196
             method), 704
                                                                                  execute() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis
enqueued_base() (AFL.automation.APIServer.Client.Client
                                                                                                method), 205
             method), 18, 153, 155
                                                                                  execute() (AFL.automation.instrument.SpecScreen_Driver.Driver
enqueued_base() (AFL.automation.loading.LoadStopperDriver.Cliemtethod), 210
             method), 234
                                                                                  execute() (AFL.automation.instrument.SpecScreen_Driver.SpecScreen_D
enqueued_base() (AFL.automation.prepare.DeckBuilderWidget.Clienethod), 28, 211, 212
                                                                                  execute() (AFL.automation.loading.LoadStopperDriver.Driver
             method), 403
enqueued_base() (AFL.automation.prepare.OT2Client.Client
                                                                                                method), 237
             method), 465
                                                                                  execute() (AFL.automation.loading.LoadStopperDriver.LoadStopperDriv
enqueued_base() (AFL.automation.prepare.OT2Client.OT2Client method), 240
                                                                                  execute() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Driv
             method), 468
enqueued_base() (AFL.automation.prepare.SampleSeriesWidget.Climethod), 258
                                                                                  \verb"execute()" (AFL. automation. loading. One Selector Blowout Sample Cell. One Selector Blowout Sample Cell
             method), 564
enqueued\_base() (AFL. automation. sample. Casting Server. Client
                                                                                                method), 261
                                                                                  execute() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Two
             method), 698
enqueued_base() (AFL.automation.sample.CastingServer.OT2Clienmethod), 266
```

830 Index

execute() (AFL.automation.loading.PneumaticPressureSampleCell.Drive

```
method), 272
                                                     FileCamera (class in AFL.automation.instrument.FileCamera),
execute() (AFL.automation.loading.PneumaticPressureSampleCell.PneumaticPressureSampleCell
                                                     FileHandler
        method), 276
execute() (AFL.automation.loading.PneumaticSampleCell.Driver AFL.automation.APIServer.APIServer), 95
        method), 282
                                                     filter() (AFL.automation.APIServer.APIServer.FileHandler
execute() (AFL.automation.loading.PneumaticSampleCell.PneumatinaSidnondNetCell
                                                     filter() (AFL.automation.APIServer.APIServer.SMTPHandler
        method), 285
execute() (AFL.automation.loading.PushPullSelectorSampleCell.Drivethod), 138
        method), 296
                                                     finalize() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
execute()(AFL.automation.loading.PushPullSelectorSampleCell.PunhPullSeleCorSampleCell
        method), 300
                                                     finalize_request() (AFL.automation.APIServer.APIServer.Flask
execute() (AFL.automation.loading.RSoXSSolutionSampleCell.Drivmethod), 116
                                                     find() (AFL.automation.shared.ServerDiscovery.AsyncZeroconfServiceTy
        method), 307
execute() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXiissolutionSdi)mpteCell
                                                     find_server_by_name()
        method), 311
execute() (AFL.automation.loading.TwoSelectorBlowoutSampleCell_AFL.automation.APIServer.Client.ServerDiscovery
        method), 352
                                                              method), 155
execute() (AFL.automation.loading.TwoSelectorBlowoutStimpdeCell/MemSeyectambel@woutSampleCell
                                                              (AFL.automation.shared.ServerDiscovery.ServerDiscovery
        method), 357
execute() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
                                                             method), 79, 764, 780
        method), 457
                                                     find_server_by_partial_name()
execute() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_QAT2_Dmixternation.APIServer.Client.ServerDiscovery
                                                              method), 155
        method), 460
execute() (AFL.automation.sample.CastingServer.CastingSirn&server_by_partial_name()
                                                              (AFL. automation. shared. Server Discovery. Server Discovery\\
        method), 695
execute() (AFL.automation.sample.CastingServer.Driver
                                                              method), 79, 764, 780
        method), 701
                                                     find_server_by_property_match()
execute() (AFL.automation.sample_env.TemperatureDeck.Driver
                                                             (AFL.automation.APIServer.Client.ServerDiscovery
        method), 707
                                                              method), 155
execute() (AFL.automation.sample_env.TemperatureDeckffixmdp.csenwerDebxk_property_match()
         method), 709
                                                              (AFL.automation.shared.ServerDiscovery.ServerDiscovery
exists() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                              method), 79, 764, 780
        method), 201
                                                     fit() (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
expanduser() (AFL.automation.instrument.SeabreezeUVVis.Path method), 718
        method), 202
                                                     fit() (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
expired_token_loader()
                                                              method), 738
        (AFL.automation.APIServer.APIServer.JWTManasat_transform() (AFL.automation.shared.DataLabelerWidget.OrdinalEn
        method), 130
                                                              method), 719
expose() (AFL.automation.instrument.DummySAS.DummySätS_transform() (AFL.automation.shared.DiffractionLabeler.OrdinalEn
                                                              method), 739
        method), 26, 186, 188
expose() (AFL.automation.instrument.122SAXS.122SAXS Flask (class in AFL.automation.APIServer,APIServer),
        method), 26, 192, 193
extensions (AFL.automation.APIServer.APIServer.Flask FLASK_DEBUG, 110
                                                     {\tt flex}\,(AFL. automation. prepare. Deck Builder Widget. Layout
        attribute), 106
extract_var() (AFL.automation.shared.DatasetWidget.DatasetWidgetribute), 432
        method), 73, 725, 731
                                                     flex
                                                             (AFL.automation.prepare.PrepareWidget.Layout
extract_var() (AFL.automation.shared.DatasetWidget.DatasetWidgetriMade) 517
        method), 73, 727, 731
                                                     flex (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                              attribute), 599
F
                                                     {\tt flex}\,(AFL. automation. prepare. Sweep Builder Widget. Layout
flex_flow(AFL.automation.prepare.DeckBuilderWidget.Layout
        attribute), 716
feature_names_in_(AFL.automation.shared.DiffractionLabeler.Ordiffableticode)2
                                                     flex_flow (AFL.automation.prepare.PrepareWidget.Layout
        attribute), 736
```

attribute			method), 557
flex_flow(AFL. attribute			(AFL.automation.prepare.SampleSeriesWidget.FloatText method), 568
			(AFL.automation.prepare.SampleSeriesWidget.HBox
attribute			method), 576
			(AFL.automation.prepare.SampleSeriesWidget.IntText
565	in AT L. automation. prepare. Sample Series	-	method), 583
FlowSelector	(class in :		
	class in : (class in : (tomation.loading.CetoniMultiPosValve		(AFL.automation.prepare.SampleSeriesWidget.Label method), 591
216	· ·		
			(AFL.automation.prepare.SampleSeriesWidget.Text
FlowSelector	(class in		method), 612
	omation.toaaing.DoubleviciMuttiposSelec		(AFL.automation.prepare.SampleSeriesWidget.VBox
227			method), 619
FlowSelector			(AFL.automation.prepare.SweepBuilderWidget.Button
	tomation.loading.FlowSelector),		method), 632
35, 232			(AFL.automation.prepare.SweepBuilderWidget.Checkbox
FlowSelector	(class in		method), 640
AFL.aut	tomation.loading. ViciMultipos Selector), = 1		(AFL.automation.prepare.SweepBuilderWidget.HBox
364			method), 647
			(AFL.automation.prepare.SweepBuilderWidget.Label
method)	• /		method), 655
			(AFL.automation.prepare.SweepBuilderWidget.Text
method)			method), 675
			(AFL.automation.prepare.SweepBuilderWidget.VBox
method)			method), 682
focus() (AFL.au method)) (AFL.automation.APIServer.APIServer.FileHandler method), 97
,) (AFL.automation.APIServer.APIServer.SMTPHandler
method)			method), 138
,			(AFL.automation.prepare.Component.Component
method)			property), 59, 382, 384
,			(AFL.automation.prepare.Solute property), 375
method)			(AFL.automation.prepare.Solvent property),
,	tomation.prepare.DeckBuilderWidget.Text		379
method)			ct() (AFL.automation.prepare.factory.Solution
	r, 443 tomation.prepare.DeckBuilderWidget.VBo		class method), 690
method)			ct() (AFL.automation.prepare.Solution class
	tomation.prepare.PrepareWidget.Button		method), 377
method)			rver_name() (AFL.automation.APIServer.Client.Client
	tomation.prepare.PrepareWidget.Checkbox		class method), 18, 152, 155
method)			rver_name() (AFL.automation.loading.LoadStopperDriver.Clien
	to mation. prepare. Prepare Widget. Dropdown		class method), 234
method)			${\tt rver_name}$ () (AFL. automation. prepare. ${\it DeckBuilderWidget}$. Client
	ıtomation.prepare.PrepareWidget.HBox		class method), 402
method)			rver_name()(AFL.automation.prepare.OT2Client.Client
focus() (AFL.au	tomation.prepare.PrepareWidget.Label		class method), 464
method)	, 510	from_se	rver_name() (AFL.automation.prepare.OT2Client.OT2Client
focus() (AFL.a	automation.prepare.PrepareWidget.Text		class method), 468
method)	, 533	from_se	${\tt rver_name}$ () (AFL. automation. prepare. Sample Series Widget. Clie
focus() (AFL.ai	utomation.prepare.PrepareWidget.VBox		class method), 563
method)	, 540	from_se	rver_name()(AFL.automation.sample.CastingServer.Client
	tomation.prepare.SampleSeriesWidget.But		class method), 698
method)			rver_name()(AFL.automation.sample.CastingServer.OT2Client
	tomation.prepare.SampleSeriesWidget.Che		

```
fromkeys() (AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.OneSelectorBlowoutSamplerCell.fletfullsdig.(AFL.automation.loading.O
                                                                                                                                                                                                                                                                                                                                                                                           class method), 266
                                                    method), 268
 fromkeys() (AFL.automation.loading.PneumaticPressureSampleCedledaulitchic) (AFL.automation.loading.PneumaticPressureSample
                                                    method), 278
                                                                                                                                                                                                                                                                                                                                                                                          class method), 271
 fromkeys() (AFL.automation.loading.PneumaticSampleCadatlefaults() (AFL.automation.loading.PneumaticPressureSample
                                                                                                                                                                                                                                                                                                                                                                                          class method), 276
                                                    method), 288
fromkeys() (AFL.automation.loading.PushPullSelectorSaraphAGedL.deftawillist() (AFL.automation.loading.PneumaticSampleCell.Dri
                                                    method), 303
                                                                                                                                                                                                                                                                                                                                                                                           class method), 281
fromkeys() (AFL.automation.loading.RSoXSSolutionSamptactaletts() (AFL.automation.loading.PneumaticSampleCell.Pne
                                                     method), 314
                                                                                                                                                                                                                                                                                                                                                                                           class method), 285
fromkeys() (AFL.automation.loading.TwoSelectorBlowout@arthpirCell.flagfutlsdigt(AFL.automation.loading.PushPullSelectorSampleC
                                                                                                                                                                                                                                                                                                                                                                                           class method), 295
                                                      method), 359
fromkeys() (AFL.automation.shared.DatasetWidget.defaultgdither_defaults() (AFL.automation.loading.PushPullSelectorSampleC
                                                                                                                                                                                                                                                                                                                                                                                           class method), 300
                                                    method), 729
Full, 743
                                                                                                                                                                                                                                                                                                                                   {\tt gather\_defaults()}\ (AFL. automation. loading. RSoXSS olution Sample Celling Sample Celling
 full_dispatch_request()
                                                                                                                                                                                                                                                                                                                                                                                           class method), 306
                                                     (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                                                                    gather_defaults() (AFL.automation.loading.RSoXSSolutionSampleCell
                                                    method), 116
                                                                                                                                                                                                                                                                                                                                                                                          class method), 311
                                                                                                                                                                                                                                                                                                                                   gather_defaults() (AFL.automation.loading.TwoSelectorBlowoutSample
G
                                                                                                                                                                                                                                                                                                                                                                                          class method), 351
\verb|gather_defaults()| (AFL. automation. APIS erver. Driver. DSA ther_defaults()| (AFL. automation. loading. Two Selector Blowout Sample and the selection of t
                                                                                                                                                                                                                                                                                                                                                                                          class method), 357
                                                      class method), 19, 158, 162
\verb|gather_defaults()| (AFL. automation. APIS erver. Dummy | \verb|gather_defaults()| (AFL. automation. prepare. Dummy | OT2 | Driver. Dri
                                                                                                                                                                                                                                                                                                                                                                                          class method), 456
                                                     class method), 164
gather_defaults()(AFL.automation.APIServer.Dummy19474epudefaplts()(AFL.automation.prepare.Dummy_OT2_Driver.Dum
                                                                                                                                                                                                                                                                                                                                                                                           class method), 461
                                                     class method), 168
{\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. 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 GFL} {\tt prodefaults()} \ (AFL. automation. sample. 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 GFL} {\tt prodefaults()} \ (AFL. automation. sample. Dummy {\tt prodefaults()} \ (AFL. automation. sample. Sa
                                                                                                                                                                                                                                                                                                                                                                                           class method), 695
                                                     class method), 171
gather_defaults()(AFL.automation.APIServer.Dummy@F2DeficedefaultysCr)veAFL.automation.sample.CastingServer.Driver
                                                                                                                                                                                                                                                                                                                                                                                           class method), 700
                                                      class method), 174
{\tt gather\_defaults()} \ (AFL. automation. instrument. Dummy {\tt SASIPSE} \ cless {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Temperature Deck. Driver {\tt faults()} \ (AFL. automation. sample\_env. Driver {\tt faults()} \ (AFL. aut
                                                                                                                                                                                                                                                                                                                                                                                           class method), 706
                                                     class method), 184
 gather_defaults()(AFL.automation.instrument.Dummy.SAS.horrndesSassIts()(AFL.automation.sample_env.TemperatureDeck.Tem
                                                                                                                                                                                                                                                                                                                                                                                           class method), 709
                                                    class method), 187
{\tt gather\_defaults()} \ (AFL. automation. instrument. I22SAX {\tt genera} {\tt te\_service\_broadcast()}
                                                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.APIServer.APIServer.Zeroconf
                                                     class method), 189
 gather_defaults()(AFL.automation.instrument.I22SAXS.I22SAXSnethod), 148
                                                                                                                                                                                                                                                                                                                                   generate_service_broadcast()
                                                     class method), 192
{\tt gather\_defaults()}\ (AFL. automation. instrument. Seabreeze UVV is. \textit{DAFF} exautomation. shared. Server Discovery. Zero configuration and the property of the property o
                                                                                                                                                                                                                                                                                                                                                                                         method), 777
                                                     class method), 195
(AFL.automation.APIServer.APIServer.Zeroconf
                                                     class method), 206
gather_defaults()(AFL.automation.instrument.SpecScreen_Driver:1910), 148
                                                                                                                                                                                                                                                                                                                                   generate_service_query()
                                                      class method), 209
{\tt gather\_defaults()} \ (AFL. automation. instrument. Spec Screen\_Driver AF Jecs upper Lippischared. Server Discovery. Zero configuration and the state of the 
                                                                                                                                                                                                                                                                                                                                                                                          method), 777
                                                     class method), 211
{\tt gather\_defaults()} \ (AFL. automation. loading. Load Stopp {\tt approximate} {\tt prupregister\_all\_services()}
                                                                                                                                                                                                                                                                                                                                                                                          (AFL.automation.APIServer.APIServer.Zeroconf
                                                      class method), 236
gather_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.LoadSt
                                                                                                                                                                                                                                                                                                                                    generate_unregister_all_services()
                                                     class method), 240
{\tt gather\_defaults()}\ (AFL. automation. loading. One Selector Blowout {\tt Samble et al. motion}, shared. Server Discovery. Zero configuration and the state of 
                                                                                                                                                                                                                                                                                                                                                                                         method), 778
                                                      class method), 257
gather_defaults()(AFL.automation.loading.OneSelectoBHOWoutSardFleCent.OneSelectoBHOWordSdrepteCelllask
                                                                                                                                                                                                                                                                                                                                                                                         method), 120
                                                     class method), 261
```

```
get() (AFL.automation.APIServer.APIServer.MutableQueue
                                                                                                                                                                                                                                                      method), 184
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.DummySAS.DummySAS
                                   method), 133
get() (AFL.automation.APIServer.Driver.PersistentConfig
                                                                                                                                                                                                                                                      method), 187
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.I22SAXS.Driver
                                   method), 161
get() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
                                                                                                                                                                                                                                                      method), 190
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.I22SAXS.I22SAXS
                                  method), 179
get() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defaulethod), 192
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.SeabreezeUVVis.Driver
                                   method), 268
get() (AFL.automation.loading.PneumaticPressureSampleCell.defauhtdibod), 195
                                                                                                                                                                                                                   \texttt{get\_config()}\ (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is.
                                   method), 278
\mathtt{get}() (AFL.automation.loading.PneumaticSampleCell.defaultdict
                                                                                                                                                                                                                                                    method), 206
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.SpecScreen_Driver.Driver
                                   method), 288
get() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdinethod), 209
                                                                                                                                                                                                                   get_config() (AFL.automation.instrument.SpecScreen_Driver.SpecScree
                                  method), 303
get() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdianethod), 211
                                   method), 314
                                                                                                                                                                                                                   get\_config() (AFL.automation.loading.LoadStopperDriver.Client
get() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defauledliod), 235
                                   method), 359
                                                                                                                                                                                                                   get_config() (AFL.automation.loading.LoadStopperDriver.Driver
get() (AFL.automation.shared.DatasetWidget.defaultdict
                                                                                                                                                                                                                                                      method), 236
                                  method), 729
                                                                                                                                                                                                                   get_config() (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.LoadSt
get() (AFL.automation.shared.MutableQueue.MutableQueue
                                                                                                                                                                                                                                                      method), 240
                                  method), 76, 742, 743
                                                                                                                                                                                                                   \verb"get_config" () (AFL. automation. loading. One Selector Blowout Sample Cell. In the configuration of the config
get() (AFL.automation.shared.PersistentConfig.MutableMapping
                                                                                                                                                                                                                                                     method), 257
                                                                                                                                                                                                                   get_config() (AFL.automation.loading.OneSelectorBlowoutSampleCell.
                                   method), 744
get() (AFL.automation.shared.PersistentConfig.PersistentConfig
                                                                                                                                                                                                                                                      method), 261
                                  method), 747
                                                                                                                                                                                                                   get_config() (AFL.automation.loading.OneSelectorBlowoutSampleCell."
get_address_and_nsec_records()
                                                                                                                                                                                                                                                      method), 266
                                  (AFL. automation. APIS erver. APIS erver. Service Infoet\_config() (AFL. automation. loading. Pneumatic Pressure Sample Cell. Description of the pressure of 
                                                                                                                                                                                                                                                      method), 271
                                  method), 143
get_address_and_nsec_records()
                                                                                                                                                                                                                   get\_config() (AFL. automation. loading. Pneumatic Pressure Sample Cell. P
                                   (AFL.automation.shared.ServerDiscovery.AsyncServiceInfomethod), 276
                                  method), 755
                                                                                                                                                                                                                   get_config() (AFL.automation.loading.PneumaticSampleCell.Driver
get_address_and_nsec_records()
                                                                                                                                                                                                                                                      method), 282
                                   (AFL.automation.shared.ServerDiscovery.Service https://onfig() (AFL.automation.loading.PneumaticSampleCell.Pneumatic
                                  method), 772
                                                                                                                                                                                                                                                      method), 285
get_components()
                                                                                           (AFL.automation.prepare.Deck get_config() (AFL.automation.loading.PushPullSelectorSampleCell.Dri
                                  method), 370
                                                                                                                                                                                                                                                      method), 295
get_composition() (AFL.automation.shared.DatasetWidgetDatasetWidgetA_IMCoahelomation.loading.PushPullSelectorSampleCell.Pus
                                  method), 74, 727, 732
                                                                                                                                                                                                                                                      method), 300
get_comps() (AFL.automation.shared.DatasetWidget.DatagetWidgetEig() (AFL.automation.loading.RSoXSSolutionSampleCell.Drive
                                  method), 73, 725, 731
                                                                                                                                                                                                                                                      method), 306
get_config() (AFL.automation.APIServer.Client.Client get_config() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX
                                  method), 18, 153, 156
                                                                                                                                                                                                                                                      method), 311
get_config() (AFL.automation.APIServer.Driver get_config() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.I
                                  method), 19, 158, 162
                                                                                                                                                                                                                                                      method), 351
get_config() (AFL.automation.APIServer.DummyDriver.Detveconfig() (AFL.automation.loading.TwoSelectorBlowoutSampleCell."
                                  method), 165
                                                                                                                                                                                                                                                      method), 357
get_config() (AFL.automation.APIServer.DummyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.DumbyDriver.Dum
                                   method), 168
                                                                                                                                                                                                                                                      method), 403
get_config() (AFL.automation.APIServer.DummyOT2Drget.Doirefig() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
                                  method), 171
                                                                                                                                                                                                                                                      method), 457
get_config() (AFL.automation.APIServer.DummyOT2Driget_Duonfix(V))(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2Driget_Duonfix(V))(AFL.automation.prepare.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.Dummy_OT2Driver.D
                                  method), 174
                                                                                                                                                                                                                                                      method), 461
get_config() (AFL.automation.instrument.DummySAS.Drgetr_config() (AFL.automation.prepare.OT2Client.Client
```

method), 465	method), 271
$\verb"get_config()" (AFL. automation. prepare. OT 2C \textit{lient}. OT 2C \textit{lient}. Config()) (AFL. automation. prepare. OT 2C \textit{lient}. OT 2C lien$	$\verb nfigs() (AFL. automation. loading. Pneumatic Pressure Sample Cell.$
method), 468	method), 276
$\verb"get_config()" (AFL. automation. prepare. Sample Series Widgest Chemother Chemother$	ndfigs() (AFL.automation.loading.PneumaticSampleCell.Driver
method), 564	method), 282
$\verb"get_config()" (AFL. automation. sample. Casting Server. Ca$	${f wfigs}$ () (AFL.automation.loading.PneumaticSampleCell.PneumaticSample and the second contract of the second
method), 695	method), 285
<pre>get_config() (AFL.automation.sample.CastingServer.Cliget_co</pre>	${\tt nfigs()}\ (AFL. automation. loading. PushPullSelector Sample Cell. Discourse the properties of th$
method), 698	method), 295
<pre>get_config() (AFL.automation.sample.CastingServer.Driget_co</pre>	nfigs() (AFL.automation.loading.PushPullSelectorSampleCell.Pi
method), 700	method), 300
get_config() (AFL.automation.sample.CastingServer.OT2)@tiento	nfigs() (AFL.automation.loading.RSoXSSolutionSampleCell.Driv
method), 704	method), 306
get_config() (AFL.automation.sample_env.TemperatureDyet_Do	
method), 707	method), 311
get_config() (AFL.automation.sample_env.TemperatureDget_Tea	
method), 709	method), 351
get_configs()(AFL.automation.APIServer.Driver.Driverget_co	
method), 19, 158, 162	method), 357
get_configs() (AFL.automation.APIServer.DummyDrivergeriveo	
method), 165	method), 457
get_configs() (AFL.automation.APIServer.DummyDriverg@tunan	
method), 168	method), 461
get_configs() (AFL.automation.APIServer.DummyOT2Dgeter.Do	
method), 171	method), 695
get_configs() (AFL.automation.APIServer.DummyOT2Dgeter.Configs())	- · ·
method), 175	method), 700
$\verb"get_configs"() (AFL. automation. instrument. Dummy SAS. \textit{Dymiter} \textbf{configs}) (AFL. automation. instrument. Dummy SAS. \textbf{Dymiter} \textbf{configs}) (AFL. automation. instrument. Dummy SAS. Dymiter) (AFL. automation. Dummy SAS. Dymiter) (AFL. automation. Instrument. Dummy SAS. Dymiter) (AFL. automation. Dummy SAS. Dymiter) (AFL.$	
method), 184	method), 707
get_configs() (AFL.automation.instrument.DummySAS.Dyentmys	
method), 187	method), 709
$\verb"get_configs()" (AFL. automation. instrument. I22SAXS. Driget_defined automation. I22SAXS. Driget_defined autom$	${\tt ck()}\ (AFL. automation. prepare. Prepare Widget. Sweep Builder Widget. Widget. Sweep Builder Widget. Widget. Sweep Builder Wid$
method), 190	method), 529
<pre>get_configs() (AFL.automation.instrument.I22SAXS.I22SpexSde</pre>	ck() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilder
method), 192	method), 68, 670, 687
get_configs()(AFL.automation.instrument.SeabreezeUV\vectDde	ek_config() (AFL.automation.prepare.DeckBuilderWidget.DeckE
method), 195	method), 60, 404, 455
get_configs()(AFL.automation.instrument.SeabreezeUV\(\) ext. Sete	okeeced(Fl/gj() (AFL.automation.prepare.PrepareWidget.DeckBuild
method), 206	method), 490
get_configs() (AFL.automation.instrument.SpecScreen_lgeite_dE	river_object()
method), 209	(AFL.automation.APIServer.APIServer.APIServer
<pre>get_configs() (AFL.automation.instrument.SpecScreen_Driver.S</pre>	
	iver_object()
get_configs() (AFL.automation.loading.LoadStopperDriver.Driv	
method), 236	method), 18, 153, 156
get_configs() (AFL.automation.loading.LoadStopperDrigertLoad	
method), 240	(AFL.automation.loading.LoadStopperDriver.Client
	* **
get_configs() (AFL.automation.loading.OneSelectorBlowoutSan	•
	iver_object()
get_configs() (AFL.automation.loading.OneSelectorBlowoutSan	
method), 261	method), 403
get_configs() (AFL.automation.loading.OneSelectorBlovgentsan	
method), 266	(AFL.automation.prepare.OT2Client.Client
<pre>get_configs() (AFL.automation.loading.PneumaticPressureSamp</pre>	oleGethAdn;v&v5

<pre>get_driver_object() (AFL.automation.prepare.OT2Client.OT2Client</pre>		(AFL.automation.prepare.SampleSeriesWidget.FloatText method), 568
	get int	eract_value()
get_driver_object()	get_Inc	(AFL.automation.prepare.SampleSeriesWidget.IntText
(AFL.automation.prepare.SampleSeriesWidget.Cl	iont	method), 583
		eract_value()
get_driver_object()	get_Inc	
		(AFL:automation.prepare.SampleSeriesWidget.Label
(AFL.automation.sample.CastingServer.Client		method), 591
	get_Int	eract_value()
get_driver_object()		(AFL:automation.prepare.SampleSeriesWidget.Text
(AFL.automation.sample.CastingServer.OT2Clier		method), 612
	get_int	eract_value()
get_feature_names_out()	. 15	(AFL:automation.prepare.SweepBuilderWidget.Checkbox
(AFL.automation.shared.DataLabelerWidget.Ord		
	get_int	eract_value()
<pre>get_feature_names_out()</pre>	1E 1	(AFL.automation.prepare.SweepBuilderWidget.Label
(AFL.automation.shared.DiffractionLabeler.Ordin		
	get_int	eract_value()
get_historical_values()		(AFL.automation.prepare.SweepBuilderWidget.Text
(AFL.automation.APIServer.Driver.PersistentCon		method), 675
		_identity() (in module
get_historical_values()		AFL.automation.APIServer.APIServer), 85
(AFL.automation.shared.PersistentConfig.Persiste method), 77, 747, 748	egett <i>ol</i> kfinga	rgs() (AFL.automation.prepare.OT2Client.PipetteAction method), 470
<pre>get_info() (AFL.automation.APIServer.APIServer.APISe</pre>	<i>r</i> gret_kwa	rgs() (AFL.automation.prepare.PipetteAction
method), 16, 92, 149		method), 373
get_interact_value()	get_lab	ware()(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_0
(AFL.automation.prepare.DeckBuilderWidget.Cha	eckbox	method), 61, 460, 462
		ager_state()
get_interact_value()		(AFL.automation.prepare.DeckBuilderWidget.Button
(AFL.automation.prepare.DeckBuilderWidget.Dro	opdown	static method), 388
	_	ager_state()
get_interact_value()	_	(AFL.automation.prepare.DeckBuilderWidget.Checkbox
(AFL.automation.prepare.DeckBuilderWidget.Lab	bel	static method), 396
		ager_state()
get_interact_value()		(AFL.automation.prepare.DeckBuilderWidget.Dropdown
(AFL.automation.prepare.DeckBuilderWidget.Tex	t	static method), 409
		ager_state()
get_interact_value()	5 –	(AFL.automation.prepare.DeckBuilderWidget.HBox
(AFL.automation.prepare.PrepareWidget.Checkbo	ox	static method), 417
		ager_state()
get_interact_value()	<i>3</i> <u>_</u>	(AFL.automation.prepare.DeckBuilderWidget.Label
(AFL.automation.prepare.PrepareWidget.Dropdo	wn	static method), 425
		ager_state()
get_interact_value()	900	(AFL.automation.prepare.DeckBuilderWidget.Layout
(AFL.automation.prepare.PrepareWidget.Label		static method), 435
	det man	ager_state()
get_interact_value()	gc c_man	(AFL.automation.prepare.DeckBuilderWidget.Text
(AFL.automation.prepare.PrepareWidget.Text		static method), 443
	net man	ager_state()
get_interact_value()	gc c_man	(AFL.automation.prepare.DeckBuilderWidget.VBox
(AFL.automation.prepare.SampleSeriesWidget.Ch	neckhov	static method), 450
		ager_state()
get interact value()	ac c_man	(AFL automation prepare PrepareWidget Rutton

static method), 478		static method), 640
<pre>get_manager_state()</pre>	get_man	ager_state()
(AFL. automation. prepare. Prepare Widget. Checklet and the content of the cont	oox	(AFL.automation.prepare.SweepBuilderWidget.HBox
static method), 485		static method), 647
<pre>get_manager_state()</pre>	get_man	ager_state()
(AFL. automation. prepare. Prepare Widget. Dropdomation and the properties of the	own	(AFL.automation.prepare.SweepBuilderWidget.Label
static method), 494		static method), 655
<pre>get_manager_state()</pre>	get_man	ager_state()
(AFL. automation. prepare. Prepare Widget. HBox		(AFL.automation.prepare.SweepBuilderWidget.Layout
static method), 502		static method), 665
<pre>get_manager_state()</pre>	get_man	ager_state()
(AFL. automation. prepare. Prepare Widget. Label		(AFL. automation. prepare. Sweep Builder Widget. Text
static method), 510		static method), 675
<pre>get_manager_state()</pre>		ager_state()
(AFL. automation. prepare. Prepare Widget. Layout		(AFL.automation.prepare.SweepBuilderWidget.VBox
static method), 520		static method), 682
<pre>get_manager_state()</pre>	get_met	adata_routing()
(AFL. automation. prepare. Prepare Widget. Text		(AFL. automation. shared. Data Labeler Widget. Ordinal Encoder
static method), 533		method), 720
<pre>get_manager_state()</pre>	get_met	adata_routing()
(AFL. automation. prepare. Prepare Widget. VBox		(AFL. automation. shared. Diffraction Labeler. Ordinal Encoder
static method), 540		method), 740
<pre>get_manager_state()</pre>	get_nam	e() (AFL.automation.APIServer.APIServer.FileHandler
(AFL.automation.prepare.SampleSeriesWidget.B	utton	method), 97
static method), 549	get_nam	e() (AFL.automation.APIServer.APIServer.ServiceInfo
<pre>get_manager_state()</pre>		method), 143
(AFL. automation. prepare. Sample Series Widget. C	<i>hgeltha</i> nam	e() (AFL.automation.APIServer.APIServer.SMTPHandler
static method), 557		method), 138
<pre>get_manager_state()</pre>	get_nam	ne() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
(AFL. automation. prepare. Sample Series Widget. F	loatText	method), 755
static method), 568	get_nam	e() (AFL.automation.shared.ServerDiscovery.ServiceInfo
<pre>get_manager_state()</pre>		method), 772
(AFL. automation. prepare. Sample Series Widget. H	Byet_non	_sample_dims()
static method), 576		$(AFL. automation. shared. Dataset Widget. Dataset Widget_Model$
<pre>get_manager_state()</pre>		method), 73, 726, 731
(AFL. automation. prepare. Sample Series Widget. In the property of the prop	<i>ıtğext_</i> obj	ect() (AFL.automation.APIServer.Client.Client
static method), 584		method), 19, 153, 156
<pre>get_manager_state()</pre>	get_obj	ect() (AFL.automation.APIServer.Driver.Driver
(AFL. automation. prepare. Sample Series Widget. Learning the series with the series of the series with the	abel	method), 20, 159, 162
static method), 591	get_obj	ect() (AFL.automation.APIServer.DummyDriver.Driver
<pre>get_manager_state()</pre>		method), 165
(AFL. automation. prepare. Sample Series Widget. Landing and the substitution of the	a ge at_obj	ect() (AFL.automation.APIServer.DummyDriver.DummyDriver
static method), 601		method), 168
<pre>get_manager_state()</pre>	get_obj	ect() (AFL.automation.APIServer.DummyOT2Driver.Driver
(AFL. automation. prepare. Sample Series Widget. To the substitution of the substitu	'ext	method), 172
static method), 612	get_obj	ect() (AFL.automation.APIServer.DummyOT2Driver.DummyDri
<pre>get_manager_state()</pre>		<i>method</i>), 175
	<i>B</i> g∧ert_obj	ect() (AFL.automation.instrument.DummySAS.Driver
static method), 619		method), 185
<pre>get_manager_state()</pre>	get_obj	ect() (AFL.automation.instrument.DummySAS.DummySAS
(AFL.automation.prepare.SweepBuilderWidget.B	Button	method), 187
static method), 632	get_obj	ect() (AFL.automation.instrument.I22SAXS.Driver
<pre>get_manager_state()</pre>		method), 190
(AFL. automation. prepare. Sweep Builder Widget. Compared to the property of	<i>ligek<u>b</u>a</i> bj	ect() (AFL.automation.instrument.I22SAXS.I22SAXS

method), 192	method), 695
<pre>get_object() (AFL.automation.instrument.SeabreezeUVVigeDribkj</pre>	ect() (AFL.automation.sample.CastingServer.Client
method), 196	method), 699
get_object() (AFL.automation.instrument.SeabreezeUVVige.Seabre	ectI(YVAFL.automation.sample.CastingServer.Driver
method), 206	method), 701
get_object() (AFL.automation.instrument.SpecScreen_Dgietr.Dbj	
method), 210	method), 704
get_object() (AFL.automation.instrument.SpecScreen_Dgetr_Species	
method), 212	method), 707
get_object() (AFL.automation.loading.LoadStopperDrivereftliobt)	
method), 235	method), 709
get_object()(AFL.automation.loading.LoadStopperDrivgenripar	
	method), 720
get_object()(AFL.automation.loading.LoadStopperDrivgetoadSt	
method), 240	method), 740
get_object() (AFL.automation.loading.OneSelectorBlowogerSaperal	
method), 258	method), 71, 712, 722
get_object() (AFL.automation.loading.OneSelectorBlowogerSapend	
	method), 75, 734, 741
get_object() (AFL.automation.loading.OneSelectorBlowogersappd	
method), 266	method), 22, 174, 175
get_object() (AFL.automation.loading.PneumaticPressugeStupple	
method), 272	method), 61, 460, 462
<pre>get_object() (AFL.automation.loading.PneumaticPressugeStangule</pre>	· · · · · · · · · · · · · · · · · · ·
method), 276	method), 16, 92, 150
get_object()(AFL.automation.loading.PneumaticSampleGetLiQues	we() (AFL.automation.APIServer.Client.Client
method), 282	method), 18, 153, 155
<pre>get_object() (AFL.automation.loading.PneumaticSampleGedLEque</pre>	սաւն)i(Ֆեներվա Թաևation.loading.LoadStopperDriver.Client
method), 285	method), 234
get_object() (AFL.automation.loading.PushPullSelectorspenpence	deDy(AFL.automation.prepare.DeckBuilderWidget.Client
method), 296	method), 403
get_object() (AFL.automation.loading.PushPullSelectorspetpleter	d k P)v.(AFill/SwlontatiSampkgCwl d.OT2Client.Client
	method), 464
get_object()(AFL.automation.loading.RSoXSSolutionSagpteQub	
	method), 468
get_object()(AFL.automation.loading.RSoXSSolutionSagpteQub	
	method), 564
get_object() (AFL.automation.loading.TwoSelectorBlow@etSaquel	
	method), 698
get_object() (AFL.automation.loading.TwoSelectorBlow@erSaquel	
	method), 704
get_object() (AFL.automation.prepare.DeckBuilderWidgge€Lique	
	(AFL.automation.APIServer.APIServer.APIServer
<pre>get_object() (AFL.automation.prepare.Dummy_OT2_Driver.Drive</pre>	
	ued_commands()
<pre>get_object() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Drive</pre>	
	method), 16, 92, 149
$\verb get_object() (AFL. automation. prepare. OT 2C lient. Client \verb get_queres) $	
method), 465	(AFL.automation.APIServer.Client.Client
	method), 18, 153, 155
	ued_commmands()
$\verb"get_object()" (AFL. automation. prepare. Sample Series Widget. Client the properties of the proper$	(AFL. automation. loading. Load Stopper Driver. Client
	method), 235
$\verb"get_object"() (AFL. automation. sample. Casting Server. Ca$	wed_commmands()

(AFL.automation.prepare.DeckBuilderWidget.Cli	ent method), 196
method), 403	<pre>get_sample() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUV</pre>
<pre>get_queued_commmands()</pre>	method), 206
(AFL. automation. prepare. OT 2Client. Client	${\tt get_sample()} \ (AFL. automation. instrument. Spec Screen_Driver. Driver$
method), 465	method), 210
<pre>get_queued_commmands()</pre>	$\verb"get_sample()" (AFL. automation. instrument. Spec Screen_Driver. Spec Screen_Driver$
(AFL. automation. prepare. OT 2Client. OT 2Client	method), 212
method), 468	<pre>get_sample() (AFL.automation.loading.LoadStopperDriver.Driver</pre>
<pre>get_queued_commmands()</pre>	method), 237
	$liget_sample()$ (AFL. automation. loading. Load Stopper Driver. Load Stopper
method), 564	method), 240
<pre>get_queued_commmands()</pre>	<pre>get_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell.</pre>
(AFL.automation.sample.CastingServer.Client	method), 258
method), 698	get_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell.
get_queued_commands()	method), 262
	nget_sample() (AFL.automation.loading.OneSelectorBlowoutSampleCell.
method), 704	method), 266
method), 16, 92, 149	Agetersample() (AFL automation.loading. Pneumatic Pressure Sample Cell. L method), 272
	memoa), 272 uget_sample() (AFL.automation.loading.PneumaticPressureSampleCell.F
method), 18, 153, 155	method), 276
	riear: Chimple() (AFL.automation.loading.PneumaticSampleCell.Driver
method), 234	method), 282
	'iglget_&imple() (AFL.automation.loading.PneumaticSampleCell.Pneumati
method), 403	method), 285
	aget_sample() (AFL.automation.loading.PushPullSelectorSampleCell.Dr
method), 465	method), 296
${\tt get_quickbar()}\ (AFL. automation. prepare. OT 2Client. OT$	${\tt 26} $ (AFL. automation. loading. PushPullSelector Sample Cell. PushPullSelector Sample Cell
method), 468	method), 300
$\verb"get_quickbar"() (AFL. automation. prepare. Sample Series)$	$ extit{MjdyetsGhiphte}()\ (AFL. automation. loading. RSoXSS olution Sample Cell. Driveton Sample Cell. Drive$
method), 564	method), 307
	Geants ample () (AFL. automation. loading. RSoXSS olution Sample Cell. RSoX
method), 698	method), 311
	QFACEample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.
method), 704	method), 352
	<pre>get_sample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.</pre>
method), 19, 159, 162	method), 357
	Detvesample() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
method), 165	method), 457 . Deabys allipilve() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O
method), 168	method), 461
	rget_Dsample() (AFL.automation.sample.CastingServer.CastingServer
method), 172	method), 695
	r get_DsamplyD(j)v(A FL.automation.sample.CastingServer.Driver
method), 175	method), 701
	getr_sample() (AFL.automation.sample_env.TemperatureDeck.Driver
method), 185	method), 707
	ngenysample() (AFL.automation.sample_env.TemperatureDeck.Temperatu
method), 187	method), 709
	$oldsymbol{aget_scattering()}$ (AFL.automation.shared.DatasetWidget.Dataset.Data
method), 190	method), 74, 727, 732
<pre>get_sample() (AFL.automation.instrument.I22SAXS.I22S</pre>	
method), 192	(AFL.automation.APIServer.APIServer.Flask
<pre>get_sample() (AFL.automation.instrument.SeabreezeUV)</pre>	Vis.Driver method), 120

```
method), 451
get_sensor_config()
             (AFL.automation.loading.PneumaticPressureSamgkeCesh.RneumukhiFPranstaneStimpheCpdre.PrepareWidget.Button
             method), 40, 277, 280
                                                                                             method), 478
get_sensor_config()
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.Checkbox
             (AFL.automation.loading.PneumaticSampleCell.PneumaticSuurlpdelCe486
             method), 42, 286, 289
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.Dropdown
get_server_time() (AFL.automation.APIServer.APIServer.APIServerthod), 495
             method), 17, 93, 151
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.HBox
get_server_time() (AFL.automation.APIServer.Client.Client
                                                                                             method), 502
                                                                                get\_state() (AFL.automation.prepare.PrepareWidget.Label
             method), 18, 153, 156
get_server_time() (AFL.automation.loading.LoadStopperDriver.Ghiethtod), 510
             method), 235
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.Layout
get_server_time() (AFL.automation.prepare.DeckBuilderWidget.Ghiethtod), 520
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.Text
             method), 403
get_server_time() (AFL.automation.prepare.OT2Client.Client
                                                                                             method), 533
             method), 465
                                                                                get_state() (AFL.automation.prepare.PrepareWidget.VBox
get_server_time() (AFL.automation.prepare.OT2Client.OT2Clientmethod), 541
             method), 468
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.Button
get_server_time()(AFL.automation.prepare.SampleSeriesWidget.Galithatd), 550
             method), 564
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.Checkbox
get_server_time() (AFL.automation.sample.CastingServer.Client method), 558
             method), 698
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.FloatText
get_server_time() (AFL.automation.sample.CastingServer.OT2Climathod), 568
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.HBox
             method), 704
get_service_info() (AFL.automation.APIServer.APIServer.Zerocon€thod), 576
             method), 146
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.IntText
get_service_info() (AFL.automation.APIServer.Client.ServerDisonvelryd), 584
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.Label
             method), 154
get_service_info() (AFL.automation.shared.ServerDiscovery.ServerDiscovery.
             method), 78, 764, 779
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.Layout
get_service_info() (AFL.automation.shared.ServerDiscovery.Zerowath$\beta$d), 601
             method), 776
                                                                                get_state() (AFL.automation.prepare.SampleSeriesWidget.Text
get_shake_latch_status()
                                                                                             method), 612
             (AFL.automation.prepare.Dummy_OT2_Driver.Dgentys_tafte_(D)(AFL.automation.prepare.SampleSeriesWidget.VBox
             method), 61, 460, 462
                                                                                             method), 620
get_shake_rpm() (AFL.automation.prepare.Dummy_OT2gDrivstDatin)vADE2udDmixenion.prepare.SweepBuilderWidget.Button
             method), 61, 460, 462
                                                                                             method), 632
get_shaker_temp() (AFL.automation.prepare.Dummy_Offet_Dstart Dummy_Offet_Dstart Dummy_off
             method), 61, 460, 462
                                                                                             method), 640
get_state() (AFL.automation.prepare.DeckBuilderWidgegButtonate() (AFL.automation.prepare.SweepBuilderWidget.HBox
             method), 389
                                                                                             method), 648
get_state() (AFL.automation.prepare.DeckBuilderWidgetet.Label
             method), 397
                                                                                             method), 656
get_state() (AFL.automation.prepare.DeckBuilderWidgetBtoptlawe() (AFL.automation.prepare.SweepBuilderWidget.Layout
             method), 410
                                                                                             method), 665
get_state() (AFL.automation.prepare.DeckBuilderWidgegHB_{CS}tate() (AFL.automation.prepare.SweepBuilderWidget.Text
             method), 417
                                                                                             method), 675
get_state() (AFL.automation.prepare.DeckBuilderWidgetyEtate() (AFL.automation.prepare.SweepBuilderWidget.VBox
             method), 425
                                                                                              method), 683
get_state() (AFL.automation.prepare.DeckBuilderWidgegetatycsttock() (AFL.automation.prepare.Deck method),
                                                                                              370
             method), 435
get_state() (AFL.automation.prepare.DeckBuilderWidgegEtxtstock_objects()
             method), 443
                                                                                             (AFL.automation.prepare.PrepareWidget.StockBuilderWidget
```

method), 528

get_state() (AFL.automation.prepare.DeckBuilderWidget.VBox

<pre>get_stock_objects()</pre>	method), 435
	ogtBuilderWidget() (AFL.automation.prepare.DeckBuilderWidget.Text
method), 66, 626, 627	method), 443
	WijdgersSionkBapidde(Wijdfet.automation.prepare.DeckBuilderWidget.VBox
method), 528	method), 451 il geri<u>V</u>ivigenSap&B(i)l(&FWidgto mation.prepare.PrepareWidget.Button
method), 66, 626, 627	т щени<u>х</u> мязем <u>яв</u>ичення тилицио таноп.prepare.Frepare wiagei.Buiton method), 478
	dgatSweispBusiplecWidAdL.automation.prepare.PrepareWidget.Checkbox
method), 529	method), 486
	memou), 400 le g\VidgiteSiwBpeRu]devtWidget omation.prepare.PrepareWidget.Dropdown
method), 68, 670, 687	method), 495
	re ge uning <u>(</u>Spe_Uni (A rF L. automation.prepare.Prepare Widget.HBox
method), 61, 460, 462	method), 503
	<pre>get_view_spec() (AFL.automation.prepare.PrepareWidget.Label</pre>
AFL.automation.shared.units), 81, 782,	method), 510
783	<pre>get_view_spec() (AFL.automation.prepare.PrepareWidget.Layout</pre>
<pre>get_unqueued_commands()</pre>	method), 520
	rget_view_spec() (AFL.automation.prepare.PrepareWidget.Text
method), 16, 92, 149	method), 533
<pre>get_unqueued_commmands()</pre>	<pre>get_view_spec() (AFL.automation.prepare.PrepareWidget.VBox</pre>
(AFL.automation.APIServer.Client.Client	method), 541
method), 18, 153, 155	$\verb"get_view_spec()" (AFL. automation. prepare. Sample Series Widget. Button with the property of the property$
<pre>get_unqueued_commmands()</pre>	method), 550
· · · · · · · · · · · · · · · · · · ·	ie $oldsymbol{v}$ $oldsymbol{t}$ (AFL. automation. prepare. Sample Series Widget. Checkles $oldsymbol{v}$
method), 234	method), 558
<pre>get_unqueued_commmands()</pre>	<pre>get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.FloatT</pre>
(AFL.automation.prepare.DeckBuilderWidget.Cl	
method), 403	<pre>get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.HBox</pre>
get_unqueued_commmands()	method), 576
(AFL.automation.prepare.OT2Client.Client	<pre>get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.IntText</pre>
method), 465	method), 584
get_unqueued_commands()	get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Label
(AFL.automation.prepare.OT2Client.OT2Client	method), 592
<pre>method), 468 get_unqueued_commmands()</pre>	<pre>get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 602</pre>
	liget_view_spec() (AFL.automation.prepare.SampleSeriesWidget.Text
method), 564	method), 612
get_unqueued_commmands()	get_view_spec() (AFL.automation.prepare.SampleSeriesWidget.VBox
(AFL.automation.sample.CastingServer.Client	method), 620
method), 698	get_view_spec() (AFL.automation.prepare.SweepBuilderWidget.Button
get_unqueued_commmands()	method), 632
	nget_view_spec() (AFL.automation.prepare.SweepBuilderWidget.Check
method), 704	method), 640
	WgielgewBewospec() (AFL.automation.prepare.SweepBuilderWidget.HBox
method), 389	method), 648
<pre>get_view_spec() (AFL.automation.prepare.DeckBuilder</pre>	Wielgew Cerackspec () (AFL.automation.prepare.SweepBuilderWidget.Label
method), 397	method), 656
<pre>get_view_spec() (AFL.automation.prepare.DeckBuilder</pre>	·W gielgew Dewpsloven () (AFL.automation.prepare.SweepBuilderWidget.Layou
method), 410	method), 666
$\verb"get_view_spec()" (AFL. automation. prepare. Deck Builder")$	WielgewildBoxspec() (AFL.automation.prepare.SweepBuilderWidget.Text
method), 418	method), 675
<pre>get_view_spec() (AFL.automation.prepare.DeckBuilder</pre>	Wyjelgen Lehrelspec () (AFL. automation. prepare. Sweep Builder Widget. VBox

method), 683

 $\verb|get_view_spec()| (AFL. automation. prepare. DeckBuilder \textit{VoietgewEdv} bas(t)) (AFL. automation. prepare. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT2_Driver. Dummy_OT3_Driver. Dummy_OT3_$

method), 425

```
method), 61, 460, 462
                                                                                                                                                                                            32, 217, 222
getChannels() (AFL.automation.loading.MultiChannelRegetPhrtain@trenklikis()
                          method), 37, 251
                                                                                                                                                                                            (AFL.automation.loading.ChemyxSyringePump.ChemyxConnecti
getChannels() (AFL.automation.loading.SainSmartRelay.MultiChannellRelay, 33, 219, 223
                          method), 318
                                                                                                                                                                getParameters() (AFL.automation.loading.ChemyxSyringePump.Chemy.
getChannels() (AFL.automation.loading.SainSmartRelay.SainSmartRelayd), 33, 219, 223
                          method), 47, 319, 320
                                                                                                                                                                getParameters() (AFL.automation.loading.PushPullSelectorSampleCell.
                                                                                                                                                                                            method), 44, 299, 304
getDisplacedVolume()
                          (AFL.automation.loading.ChemyxSyringePump.ChemyxCth)deAtoth.automation.loading.CetoniMultiPosValve.CetoniMultiPos
                          method), 33, 219, 223
                                                                                                                                                                                            method), 31, 215, 216
getElapsedTime() (AFL.automation.loading.ChemyxSyringertPhonp.(C)henFlx:Qutmnactiionn.loading.CetoniMultiPosValve.FlowSelector
                          method), 33, 219, 223
                                                                                                                                                                                            method), 216
getExposure() (AFL.automation.instrument.SeabreezeUV\sex Beatre\chi (AFL) \text{Mintomation.loading.DoubleViciMultiposSelector.DoubleV
                          method), 27, 205, 207
                                                                                                                                                                                            method), 34, 227, 229
getExposureDelay() (AFL.automation.instrument.SeabregeatP&Fis(SeaHFLequL&MVitson.loading.DoubleViciMultiposSelector.FlowSel
                          method), 27, 205, 207
                                                                                                                                                                                            method), 227
getFilename() (AFL.automation.instrument.SeabreezeUVWexSeabvextAFVXintomation.loading.DoubleViciMultiposSelector.ViciMult
                          method), 27, 205, 207
                                                                                                                                                                                            method), 228
getFilepath() (AFL.automation.instrument.SeabreezeUVgetBeath() (AFL.automation.loading.FlowSelector.FlowSelector
                          method), 27, 205, 207
                                                                                                                                                                                            method), 35, 232
getLevel() (AFL.automation.loading.ChemyxSyringePumg.ctRorrx(SyriNgERumpmation.loading.ViciMultiposSelector.FlowSelector
                          method), 32, 220, 222
                                                                                                                                                                                            method), 364
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), 57, 366
                          method), 135
getName() (AFL.automation.APIServer.QueueDaemon.QuegetPump&tatus() (AFL.automation.loading.ChemyxSyringePump.Chemy.
                          method), 181
                                                                                                                                                                                            method), 33, 219, 223
getName() (AFL.automation.EpicsADLiveProcess.ReduceDyearBart ReduAHDacentomation.loading.ChemyxSyringePump.ChemyxSyring
                                                                                                                                                                                            method), 32, 220, 222
                          method), 795
getName() (AFL.automation.loading.LoadStopperDriver.SagetRading(AlFladutomation.loading.ChemyxSyringePump.SyringePump
                          method), 243
                                                                                                                                                                                            method), 221
getName() (AFL.automation.loading.LoadStopperDriver.SigetRad&By(AFL.automation.loading.DummyPump.DummyPump
                          method), 246
                                                                                                                                                                                            method), 35, 230, 231
getName() (AFL.automation.loading.LoadStopperDriver.StgetRatleBy2AFL.automation.loading.DummyPump.SyringePump
                          method), 249
                                                                                                                                                                                            method), 231
getName() (AFL.automation.loading.Sensor.DummySensorgetRate() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePum
                                                                                                                                                                                           method), 38, 254, 255
                          method), 323
getName() (AFL.automation.loading.Sensor.DummySensorgetRate() (AFL.automation.loading.NE1kSyringePump.SyringePump
                          method), 325
                                                                                                                                                                                            method), 255
getName() (AFL.automation.loading.SensorCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThreagestrateCallbackThr
                                                                                                                                                                                            method), 43, 292, 294
                          method), 331
getName() (AFL.automation.loading.SensorCallbackThreadeSiBarteQ)reAlFaldaGBomation.loading.PressureControllerAsPump.SyringeP
                          method), 333
                                                                                                                                                                                            method), 293
getName() (AFL.automation.loading.SensorCallbackThreageStBaltedQCBFL.automation.loading.SyringePump.SyringePump
                                                                                                                                                                                            method), 54, 348
                          method), 336
getName() (AFL.automation.loading.SensorCallbackThreages\Belspond(SB()) (AFL.automation.loading.ChemyxSyringePump.ChemyxC
                          method), 339
                                                                                                                                                                                            method), 33, 219, 222
getName() (AFL.automation.loading.SensorPollingThread.get.SavPeslingTheScrath()
                                                                                                                                                                                            (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is
                          method), 345
{\tt getName()} \ (AFL. automation. shared. Server Discovery. Run Thread
                                                                                                                                                                                            method), 27, 205, 207
                          method), 762
                                                                                                                                                                getServerState() (AFL. automation. loading. Sensor Callback Thread. Loading. Sensor Callback Thr
getName() (AFL.automation.shared.ServerDiscovery.ServiceBrowsemethod), 52, 329, 343
                                                                                                                                                                 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
                          method), 767
getOpenPorts()
                                                                                        (in
                                                                                                                                       module
                                                                                                                                                                                            method), 32, 221, 222
```

AFL.automation.loading.ChemyxSyringePump), getStatus() (AFL.automation.loading.DummyPump.DummyPump

```
method), 35, 230, 231
                                                      grid_column(AFL.automation.prepare.PrepareWidget.Layout
getStatus() (AFL.automation.loading.NE1kSyringePump.NE1kSyringtePounter), 519
        method), 38, 254, 256
                                                      grid_column(AFL.automation.prepare.SampleSeriesWidget.Layout
getStatus()(AFL.automation.loading.PressureControllerAsPump.RnesibureGonoRollerAsPump
         method), 43, 292, 294
                                                       grid\_column(AFL.automation.prepare.SweepBuilderWidget.Layout
getSubject() (AFL.automation.APIServer.APIServer.SMTPHandlerattribute), 664
        method), 137
                                                      grid_gap (AFL.automation.prepare.DeckBuilderWidget.Layout
getValueFromParams()
                                                                attribute), 433
         (AFL.automation.loading.ChemyxSyringePump.Charindx.Syrin(AelFluomptomation.prepare.PrepareWidget.Layout
                                                                attribute), 518
        method), 32, 221, 222
glob() (AFL.automation.instrument.SeabreezeUVVis.Path grid_gap(AFL.automation.prepare.SampleSeriesWidget.Layout
         method), 200
                                                                attribute), 600
got_first_request (AFL.automation.APIServer.APIServer.Edustary (AFL.automation.prepare.SweepBuilderWidget.Layout
        property), 107
                                                                attribute), 664
goto_callback() (AFL.automation.shared.DataLabelerWgried_DwwLAbdlanWodgettion.prepare.DeckBuilderWidget.Layout
         method), 71, 714, 721
                                                                attribute), 434
goto_callback() (AFL.automation.shared.DatasetWidgetqPattusetWildPelL.automation.prepare.PrepareWidget.Layout
         method), 73, 725, 731
                                                                attribute), 519
goto_callback() (AFL.automation.shared.DiffractionLabgleinDiffractAbilLabalmation.prepare.SampleSeriesWidget.Layout
         method), 75, 733, 741
                                                                attribute), 600
grid_area (AFL.automation.prepare.DeckBuilderWidget.Lgyard_row (AFL.automation.prepare.SweepBuilderWidget.Layout
         attribute), 434
                                                                attribute), 664
grid_area(AFL.automation.prepare.PrepareWidget.Layougrid_template_areas
         attribute), 519
                                                                (AFL.automation.prepare.DeckBuilderWidget.Layout
grid\_area\ (AFL. automation. prepare. Sample Series Widget. Layout
                                                                attribute), 433
         attribute), 600
                                                       grid_template_areas
{\tt grid\_area} (AFL. automation. prepare. Sweep Builder Widget. Layout
                                                                (AFL.automation.prepare.PrepareWidget.Layout
                                                                attribute), 518
        attribute), 664
grid_auto_columns(AFL.automation.prepare.DeckBuild@pt@thdgtehpladte_areas
         attribute), 433
                                                                (AFL.automation.prepare.SampleSeriesWidget.Layout
grid_auto_columns (AFL.automation.prepare.PrepareWidget.Layouttribute), 600
         attribute), 518
                                                       grid_template_areas
grid_auto_columns (AFL.automation.prepare.SampleSeriesWidget.Layout
                                                                attribute), 664
         attribute), 600
grid_auto_columns (AFL.automation.prepare.SweepBuildparWidgemblactur_columns
                                                                (AFL.automation.prepare.DeckBuilderWidget.Layout
        attribute), 664
\verb|grid_auto_flow| (AFL. automation. prepare. Deck Builder Widget. Layo \textit{\textbf{at}} tribute), 433
         attribute), 433
                                                       grid_template_columns
grid_auto_flow (AFL.automation.prepare.PrepareWidget.Layout (AFL.automation.prepare.PrepareWidget.Layout
         attribute), 518
                                                                attribute), 518
grid_auto_flow (AFL.automation.prepare.SampleSeriesWgdiget_Itaennpltate_columns
         attribute), 600
                                                                (AFL.automation.prepare.SampleSeriesWidget.Layout
grid_auto_flow(AFL.automation.prepare.SweepBuilderWidget.Layattribute), 600
                                                       grid_template_columns
         attribute), 664
grid_auto_rows (AFL.automation.prepare.DeckBuilderWidget.LayoutAFL.automation.prepare.SweepBuilderWidget.Layout
         attribute), 433
                                                                attribute), 664
grid_auto_rows (AFL.automation.prepare.PrepareWidgetdxixdytemplate_rows (AFL.automation.prepare.DeckBuilderWidget.Layo
         attribute), 518
                                                                attribute), 433
grid_auto_rows (AFL.automation.prepare.SampleSeriesWgdget_Itaepmpltate_rows (AFL.automation.prepare.PrepareWidget.Layout
         attribute), 600
                                                                attribute), 518
```

attribute), 664

attribute), 434

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), 600

attribute), 664

group()) (AFL.automation.instrument.SeabreezeUVVis.Pat method), 200		static method), 478 comm_opened()
Н			(AFL.automation.prepare.PrepareWidget.Checkbox static method), 486
halt()	(AFL.automation.APIServer.APIServer.APIServer method), 17, 93, 150	handle_	_comm_opened() (AFL.automation.prepare.PrepareWidget.Dropdown
halt()	(AFL.automation.APIServer.Client.Client method), 18, 153, 156	handle_	static method), 495 .comm_opened()
halt()	(AFL.automation.EpicsADLiveProcess.Client.Client.method), 24, 792, 793		(AFL.automation.prepare.PrepareWidget.HBox static method), 503
halt()	(AFL.automation.loading.LoadStopperDriver.Clientethod), 235	\imath_l handle_	
halt()	(AFL.automation.prepare.DeckBuilderWidget.Clientethod), 403		static method), 510 comm_opened()
halt()	(AFL.automation.prepare.OT2Client.Client method), 465		(AFL.automation.prepare.PrepareWidget.Layout static method), 520
halt()	(AFL.automation.prepare.OT2Client.OT2Client method), 468	handle_	.comm_opened() (AFL.automation.prepare.PrepareWidget.Text
halt()	(AFL.automation.prepare.SampleSeriesWidget.Clie method), 564		static method), 533 .comm_opened()
halt()	(AFL.automation.sample.CastingServer.Client method), 698	_	(AFL.automation.prepare.PrepareWidget.VBox static method), 541
halt()	(AFL.automation.sample.CastingServer.OT2Client method), 704	handle_	
handle	() (AFL.automation.APIServer.APIServer.FileHand method), 97		static method), 550 comm_opened()
handle	() (AFL.automation.APIServer.APIServer.SMTPHa method), 138		(AFL.automation.prepare.SampleSeriesWidget.Checkbox static method), 558
handle	_comm_opened()	handle_	_comm_opened()
	(AFL.automation.prepare.DeckBuilderWidget.Bu static method), 389		(AFL.automation.prepare.SampleSeriesWidget.FloatText static method), 568
handle	_comm_opened()	handle_	_comm_opened()
iidiidic_	(AFL.automation.prepare.DeckBuilderWidget.Ch static method), 397		(AFL.automation.prepare.SampleSeriesWidget.HBox static method), 576
handle	_comm_opened()	handle_	_comm_opened()
	(AFL.automation.prepare.DeckBuilderWidget.Dr static method), 410	opdown	(AFL.automation.prepare.SampleSeriesWidget.IntText static method), 584
handle	_comm_opened()	handle_	_comm_opened()
	(AFL.automation.prepare.DeckBuilderWidget.HE static method), 418	Вох	(AFL.automation.prepare.SampleSeriesWidget.Label static method), 592
handle_	_comm_opened() (AFL.automation.prepare.DeckBuilderWidget.La		_comm_opened() (AFL.automation.prepare.SampleSeriesWidget.Layout
handle_	static method), 425 _comm_opened()		static method), 602 _comm_opened()
	(AFL.automation.prepare.DeckBuilderWidget.La static method), 435	yout	(AFL.automation.prepare.SampleSeriesWidget.Text static method), 612
handle_	_comm_opened()	handle_	comm_opened()
	(AFL.automation.prepare.DeckBuilderWidget.Textstatic method), 443	xt	(AFL.automation.prepare.SampleSeriesWidget.VBox static method), 620
handle_	_comm_opened()	handle_	_comm_opened()
_	(AFL.automation.prepare.DeckBuilderWidget.VB static method), 451		(AFL.automation.prepare.SweepBuilderWidget.Button static method), 633
handle_	_comm_opened()	handle_	_comm_opened()
	(AFL.automation.prepare.PrepareWidget.Button		(AFL. automation. prepare. Sweep Builder Widget. Checkbox

static method), 640	class method), 510
	andle_control_comm_opened()
(AFL.automation.prepare.SweepBuilderWidget.HB	
static method), 648	class method), 520
	andle_control_comm_opened()
(AFL. automation. prepare. Sweep Builder Widget. Laber 1990 and 1990 and 1990 are the properties of	
static method), 656	class method), 533
handle_comm_opened()	andle_control_comm_opened()
(AFL.automation.prepare.SweepBuilderWidget.Lay	
static method), 666	class method), 541
	andle_control_comm_opened()
(AFL.automation.prepare.SweepBuilderWidget.Text	
static method), 675	class method), 550
handle_comm_opened() h	andle_control_comm_opened()
(AFL. automation. prepare. Sweep Builder Widget. VB and the property of the	ox (AFL.automation.prepare.SampleSeriesWidget.Checkbox
static method), 683	class method), 558
<pre>handle_control_comm_opened()</pre>	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.Butto	on (AFL.automation.prepare.SampleSeriesWidget.FloatText
class method), 389	class method), 568
<pre>handle_control_comm_opened()</pre>	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.Chec	
class method), 397	class method), 576
	andle_control_comm_opened()
	down (AFL.automation.prepare.SampleSeriesWidget.IntText
class method), 410	class method), 584
	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.HBo	
class method), 418	class method), 592
	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.Labe	
class method), 425	class method), 602
	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.Layo	
class method), 435	class method), 612
	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.Text	
class method), 443	class method), 620
	andle_control_comm_opened()
(AFL.automation.prepare.DeckBuilderWidget.VBox	1 1
class method), 451	class method), 633
	andle_control_comm_opened()
(AFL.automation.prepare.PrepareWidget.Button	(AFL.automation.prepare.SweepBuilderWidget.Checkbo.
class method), 478	class method), 640
	andle_control_comm_opened()
(AFL. automation. prepare. Prepare Widget. Checkbox	
class method), 486	class method), 648
	andle_control_comm_opened()
(AFL. automation. prepare. Prepare Widget. Dropdown	
class method), 495	class method), 656
	andle_control_comm_opened()
(AFL. automation. prepare. Prepare Widget. HBox	(AFL. automation. prepare. Sweep Builder Widget. Layout
class method), 503	class method), 666
handle_control_comm_opened() h	andle_control_comm_opened()
(AFL.automation.prepare.PrepareWidget.Label	(AFL. automation. prepare. Sweep Builder Widget. Text

class method), 675	1	nethod),	541			
handle_control_comm_opened()	has_trai	t()(AF	L.automatio	n.prepare.San	npleSeriesWid	lget.Button
(AFL. automation. prepare. Sweep Builder Widget. V. automation. Prepare. Pr	Box 1	nethod),	550			
class method), 683	has_trai	t()(AF	L.automatio	n.prepare.San	npleSeriesWid	lget.Checkbox
handle_exception() (AFL.automation.APIServer.APISe	rver.Flask r	nethod),	558			
method), 115	has_trai	t()(AF	L.automatio	n.prepare.San	npleSeriesWid	lget.FloatText
handle_http_exception()		nethod),			•	
(AFL.automation.APIServer.APIServer.Flask				n.prepare.San	npleSeriesWid	lget.HBox
method), 114		nethod),		1 1	1	Ö
handle_url_build_error()				n.prepare.San	npleSeriesWid	lget.IntText
(AFL.automation.APIServer.APIServer.Flask		nethod),		. F	T	
method), 124				n.prepare.San	nnleSeriesWid	lget.Lahel
handle_user_exception()		nethod),		n.prepare.san	ipreseries (rie	18ст. Дагост
(AFL.automation.APIServer.APIServer.Flask				n.prepare.San	nnleSeriesWie	laet Lavout
method), 115		nethod),		п.ргераге.ван	ipieseries wie	ідсі.Диубиі
handleError() (AFL.automation.APIServer.APIServer.Fi				n nranara San	anla Carias Wid	last Taxt
method), 97		nethod),		п.ргераге.зап	ipieserieswii	igei.1exi
				Ca	an LaCani aa Wi	doot VD ou
handleError() (AFL.automation.APIServer.APIServer.SM				n.prepare.san	ıpıeserieswic	iget. v Box
method), 138		nethod),		G.	D :11 HV:	1 . D
hardlink_to() (AFL.automation.instrument.SeabreezeUV				n.prepare.Swe	гервинаегын	aget.Button
method), 201		nethod),		a	D 111 YY	
has_static_folder(AFL.automation.APIServer.APIServ				n.prepare.Swe	epBuilderWi	dget.Checkbox
property), 120		nethod),				
$\verb has_trait() (AFL. automation. prepare. Deck Builder Widge Grant For the Comparison of the Compari$				n.prepare.Swe	epBuilderWi	dget.HBox
method), 389		nethod),				
<pre>has_trait() (AFL.automation.prepare.DeckBuilderWidge</pre>		t() (AF)		n.prepare.Swe	epBuilderWi	dget.Label
has_trait()(AFL.automation.prepare.DeckBuilderWidge				n.prepare.Swe	epBuilderWi	dget.Lavout
method), 410	•	nethod),		1 1	1	0 ,
has_trait() (AFL.automation.prepare.DeckBuilderWidge				n.prepare.Swe	epBuilderWi	dget.Text
method), 418		nethod),		p. ep al elemen	ep 2 mae. Tra	
has_trait() (AFL.automation.prepare.DeckBuilderWidge				n prepare Swe	enRuilderWi	doet VRox
method), 425		nethod),		п.ргераге.вис	ервинаетт	agei. v Box
has_trait() (AFL.automation.prepare.DeckBuilderWidge				(in	module	
method), 435	•			oare.factory),		
has_trait() (AFL.automation.prepare.DeckBuilderWidge						
1 1				automation.s	narea.umiis),	
method), 443		30, 782,		(:	11 -	
has_trait() (AFL.automation.prepare.DeckBuilderWidge				(in	module	
method), 451				red.utilities), 7		٨
has_trait()(AFL.automation.prepare.PrepareWidget.Bu			.automation	.prepare.Deck	BuilderWidge	et),
method), 478		114		D	****	
has_trait()(AFL.automation.prepare.PrepareWidget.Ch			L.automatio	n.prepare.Pre	pareWidget),	
method), 486		199	_	~		
has_trait() (AFL.automation.prepare.PrepareWidget.Dr method), 495		s in AFL 573	.automation	.prepare.Samp	oleSeriesWidg	get),
has_trait()(AFL.automation.prepare.PrepareWidget.HB	BHBox (clas	s in AFL	.automation	.prepare.Swee	pBuilderWids	get).
method), 503		544				, ,,
has_trait()(AFL.automation.prepare.PrepareWidget.La	が約20Fact	orv() ((in module	AFL automati	ion.prepare).	
method), 510		367			,,	
has_trait() (AFL.automation.prepare.PrepareWidget.La	-			(in	module	
method), 520	•	-	omation pre	pare.factory),	68, 688,	
has_trait() (AFL.automation.prepare.PrepareWidget.Te.		11 L.aut 592	лишион.ргеј	sare.juciory),	00, 000,	
method), 533			mation prop	are.DeckBuila	lorWidget I a	vout
has_trait() (AFL.automation.prepare.PrepareWidget.VE		r L.auio ittribute		ате.Бесквини	a mugei.Lu	oni
nas_crarc() (Ar L.uniomunon.prepare.Freparewiaget.VE	ιοι (un wuie,	J, → J∠			

```
height (AFL.automation.prepare.PrepareWidget.Layout hold_sync() (AFL.automation.prepare.SampleSeriesWidget.VBox
        attribute), 517
                                                             method), 620
height (AFL.automation.prepare.SampleSeriesWidget.Laydnold_sync() (AFL.automation.prepare.SweepBuilderWidget.Button
        attribute), 599
                                                             method), 633
height (AFL.automation.prepare.SweepBuilderWidget.Laydmatld_sync() (AFL.automation.prepare.SweepBuilderWidget.Checkbox
                                                             method), 641
        attribute), 663
hold_sync() (AFL.automation.prepare.DeckBuilderWidgehBldgsync() (AFL.automation.prepare.SweepBuilderWidget.HBox
        method), 389
                                                              method), 648
hold_sync() (AFL.automation.prepare.DeckBuilderWidgeh6heckbync() (AFL.automation.prepare.SweepBuilderWidget.Label
        method), 397
                                                             method), 656
hold_sync() (AFL.automation.prepare.DeckBuilderWidgehDrdpsynne() (AFL.automation.prepare.SweepBuilderWidget.Layout
        method), 410
                                                             method), 666
hold_sync() (AFL.automation.prepare.DeckBuilderWidgehbfBbxsync() (AFL.automation.prepare.SweepBuilderWidget.Text
        method), 418
                                                             method), 675
hold_sync() (AFL.automation.prepare.DeckBuilderWidgehbalde/sync() (AFL.automation.prepare.SweepBuilderWidget.VBox
        method), 425
                                                             method), 683
hold_sync() (AFL.automation.prepare.DeckBuilderWidgehbdqoptrait_notifications()
        method), 435
                                                             (AFL.automation.prepare.DeckBuilderWidget.Button
                                                             method), 389
hold_sync() (AFL.automation.prepare.DeckBuilderWidget.Text
        method), 443
                                                    hold_trait_notifications()
hold_sync() (AFL.automation.prepare.DeckBuilderWidget.VBox
                                                             (AFL.automation.prepare.DeckBuilderWidget.Checkbox
        method), 451
                                                             method), 397
hold_sync() (AFL.automation.prepare.PrepareWidget.Buthold_trait_notifications()
                                                              (AFL.automation.prepare.DeckBuilderWidget.Dropdown
        method), 478
                                                             method), 410
hold_sync() (AFL.automation.prepare.PrepareWidget.Checkbox
        method), 486
                                                    hold_trait_notifications()
hold_sync() (AFL.automation.prepare.PrepareWidget.Dropdown
                                                             (AFL.automation.prepare.DeckBuilderWidget.HBox
                                                             method), 418
        method), 495
hold_sync() (AFL.automation.prepare.PrepareWidget.HBhold_trait_notifications()
        method), 503
                                                              (AFL.automation.prepare.DeckBuilderWidget.Label
hold_sync() (AFL.automation.prepare.PrepareWidget.Label
                                                             method), 425
        method), 510
                                                    hold_trait_notifications()
                                                             (AFL.automation.prepare.DeckBuilderWidget.Layout
hold_sync() (AFL.automation.prepare.PrepareWidget.Layout
                                                             method), 435
        method), 520
hold_sync()(AFL.automation.prepare.PrepareWidget.Texhold_trait_notifications()
                                                             (AFL.automation.prepare.DeckBuilderWidget.Text
        method), 533
hold_sync() (AFL.automation.prepare.PrepareWidget.VBox
                                                             method), 443
        method), 541
                                                    hold_trait_notifications()
hold_sync() (AFL.automation.prepare.SampleSeriesWidget.Button (AFL.automation.prepare.DeckBuilderWidget.VBox
                                                             method), 451
        method), 550
hold_sync() (AFL.automation.prepare.SampleSeriesWidgenotheathait_notifications()
        method), 558
                                                              (AFL.automation.prepare.PrepareWidget.Button
hold_sync() (AFL.automation.prepare.SampleSeriesWidget.FloatTextethod), 478
                                                    hold_trait_notifications()
        method), 569
hold_sync() (AFL.automation.prepare.SampleSeriesWidget.HBox (AFL.automation.prepare.PrepareWidget.Checkbox
        method), 576
                                                             method), 486
hold_sync() (AFL.automation.prepare.SampleSeriesWidg&rolldTetwait_notifications()
                                                             (AFL.automation.prepare.PrepareWidget.Dropdown
        method), 584
hold_sync() (AFL.automation.prepare.SampleSeriesWidget.Label method), 495
        method), 592
                                                    hold_trait_notifications()
hold_sync() (AFL.automation.prepare.SampleSeriesWidget.Layout (AFL.automation.prepare.PrepareWidget.HBox
        method), 602
                                                             method), 503
hold_sync() (AFL.automation.prepare.SampleSeriesWidgtoTedttrait_notifications()
```

method), 612

(AFL.automation.prepare.PrepareWidget.Label

method), 510		method), 675
hold_trait_notifications()		rait_notifications()
(AFL.automation.prepare.PrepareWidget.Layout method), 520		(AFL.automation.prepare.SweepBuilderWidget.VBox method), 683
hold_trait_notifications()	home()	(AFL.automation.EpicsADLiveProcess.Client.Client
(AFL.automation.prepare.PrepareWidget.Text		method), 24, 792, 793
method), 533	home()	(AFL.automation.instrument.SeabreezeUVVis.Path
hold_trait_notifications()		class method), 199
(AFL.automation.prepare.PrepareWidget.VBox method), 541	home()	(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Driver.dummy_OT2_Driver.Dummy_OT2_Driver
hold_trait_notifications()	home()	(AFL.automation.prepare.OT2Client.OT2Client
(AFL. automation. prepare. Sample Series Widget. Between the substitution of the sub		method), 62, 468, 470
method), 550	home()	(AFL.automation.sample.CastingServer.OT2Client
hold_trait_notifications()		method), 703
(AFL.automation.prepare.SampleSeriesWidget.C method), 558	<i>hlaolsh</i> toxtt	£1 (AFL.automation.APIServer.APIServer.ServiceInfo attribute), 141
hold_trait_notifications()	host_tt	al (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
(AFL.automation.prepare.SampleSeriesWidget.F.		
method), 569	host_tt	1 (AFL.automation.shared.ServerDiscovery.ServiceInfo
hold_trait_notifications()	rn.	attribute), 770
	<i>H</i> now_mar	ny() (AFL.automation.APIServer.DummyDriver.DummyDriver
method), 576	1	method), 21, 168, 169
hold_trait_notifications()		ny () (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
(AFL.automation.prepare.SampleSeriesWidget.In method), 584	u 1ext	method), 22, 174, 176
hold_trait_notifications()	ı	
(AFL.automation.prepare.SampleSeriesWidget.Lo method), 592	a h2 2SAXS	S (class in AFL.automation.instrument.I22SAXS), 26, 191, 193
hold_trait_notifications()	icon(Al	FL.automation.prepare.DeckBuilderWidget.Button
(AFL. automation. prepare. Sample Series Widget. Loss and the series with th	ayout	attribute), 387
method), 602	icon(A)	FL.automation.prepare.PrepareWidget.Button at-
hold_trait_notifications()		tribute), 476
(AFL.automation.prepare.SampleSeriesWidget.Temperature (AFL.automation.prepare.SampleSeriesWidget.Temperature), 612	^{?X} fcon (AI	FL.automation.prepare.SampleSeriesWidget.Button attribute), 548
hold_trait_notifications()	icon(A)	FL.automation.prepare.SweepBuilderWidget.Button
(AFL. automation. prepare. Sample Series Widget. V. automation. Prepare. Prepar	Box	attribute), 631
method), 620	ident(A	AFL.automation.APIServer.APIServer.QueueDaemon
hold_trait_notifications()		property), 135
(AFL. automation. prepare. Sweep Builder Widget. But the substitution of the substit	<i>ਘ1t</i> ውent (A	AFL.automation.APIServer.QueueDaemon.QueueDaemon
method), 633		property), 181
hold_trait_notifications()		$\Lambda FL. automation. Epics ADL ive Process. Reduce Daemon. Reduce Daemon and the process of the p$
(AFL. automation. prepare. Sweep Builder Widget. Color for the property of t	'heckbox	property), 795
method), 641	ident(A	AFL. automation. loading. Load Stopper Driver. Sensor Polling Thread
hold_trait_notifications()	**	property), 243
(AFL.automation.prepare.SweepBuilderWidget.H	Bident (A	AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
method), 648		property), 246
hold_trait_notifications()	ident(A	AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
(AFL.automation.prepare.SweepBuilderWidget.L		property), 249
method), 656	ident ((AFL.automation.loading.Sensor.DummySensor1
hold_trait_notifications()		property), 323
(AFL.automation.prepare.SweepBuilderWidget.L method), 666	<i>а</i> Үвё ht ((AFL.automation.loading.Sensor.DummySensor2 property), 325
hold_trait_notifications()	ident(/	AFL.automation.loading.SensorCallbackThread.SensorCallbackTh
(AFL. automation. prepare. Sweep Builder Widget. The property of the propert	ext	property), 331

```
ident (AFL.automation.loading.SensorCallbackThread.SimpleThresholdtatl), 82, 695, 705
                                                                                                                                                                                                                                 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.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.DatasetWidge
                                    property), 333
ident (AFL.automation.loading.SensorCallbackThread.StopLoadCBvMethod), 74, 728, 732
                                                                                                                                                                                                                                 init_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget
                                    property), 336
ident (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2ethod), 74, 728, 732
                                                                                                                                                                                                                                 init_inputs() (AFL.automation.shared.DatasetWidget.DatasetWidget_V
                                    property), 339
ident (AFL.automation.loading.SensorPollingThread.SensorPollingThmethdd), 74, 728, 732
                                                                                                                                                                                                                                 init_logging() (AFL.automation.APIServer.APIServer.APIServer
                                    property), 345
ident(AFL. automation. shared. Server Discovery. Run Thread
                                                                                                                                                                                                                                                                       method), 16, 92, 149
                                    property), 762
                                                                                                                                                                                                                                 init_models() (AFL.automation.shared.DataLabelerWidget.DataLabeler
{\tt ident}\, (AFL. automation. shared. Server Discovery. Service Browser
                                                                                                                                                                                                                                                                      method), 71, 712, 722
                                                                                                                                                                                                                                  init_models() (AFL.automation.shared.DiffractionLabeler.DiffractionLa
                                    property), 767
import\_name\ (AFL. automation. APIS erver. APIS erver. Flask
                                                                                                                                                                                                                                                                       method), 75, 734, 741
                                                                                                                                                                                                                                 init_plots() (AFL.automation.shared.DatasetWidget.DatasetWidget_Vie
                                    attribute), 123
                                                                                                                                                                                                                                                                       method), 74, 728, 732
in_bounds()
                                                                (AFL.automation.prepare.MassBalance
                                     method), 372
                                                                                                                                                                                                                                 init_remote_connection()
indent (AFL.automation.prepare.DeckBuilderWidget.Checkbox
                                                                                                                                                                                                                                                                       (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                                                                                        method),
                                     attribute), 395
indent(AFL. automation. prepare. PrepareWidget. Checkboxinitialize\_plots()(AFL. automation. shared. DatasetWidget. DatasetWi
                                     attribute), 484
                                                                                                                                                                                                                                                                       method), 73, 725, 731
indent(AFL.automation.prepare.SampleSeriesWidget.Checkbyect_url_defaults()
                                     attribute), 556
                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.APIServer.Flask
{\tt indent}\, (AFL. automation. prepare. Sweep Builder Widget. Checkbox
                                                                                                                                                                                                                                                                      method), 120
                                                                                                                                                                                                                                 instance\_path(AFL.automation.APIServer.APIServer.Flask
                                      attribute), 638
index (AFL.automation.prepare.DeckBuilderWidget.Dropdown
                                                                                                                                                                                                                                                                       attribute), 105
                                     attribute), 410
                                                                                                                                                                                                                                 interface\_index (AFL. automation. APIS erver. APIS erver. Service Info
index (AFL. automation. prepare. Prepare Widget. Dropdown
                                                                                                                                                                                                                                                                       attribute), 141
                                     attribute), 495
                                                                                                                                                                                                                                 \verb"interface_index" (AFL. automation. shared. Server Discovery. Async Service Service
index() (AFL.automation.APIServer.APIServer.APIServer
                                                                                                                                                                                                                                                                       attribute), 753
                                     method), 16, 92, 150
                                                                                                                                                                                                                                 interface_index(AFL.automation.shared.ServerDiscovery.ServiceInfo
\verb"index_new"() (AFL. automation. APIS erver. APIS er
                                                                                                                                                                                                                                                                       attribute), 770
                                    method), 16, 92, 150
                                                                                                                                                                                                                                 IntText (class in AFL.automation.prepare.SampleSeriesWidget),
infrequent_categories_
                                     (AFL.automation.shared.DataLabelerWidget.Ordinnt/Enlandletoken_loader()
                                    attribute), 716
                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.APIServer.JWTManager
infrequent_categories_
                                                                                                                                                                                                                                                                       method), 131
                                    (AFL.automation.shared.DataLabelerWidget.OrdinnWennseletransform()
                                                                                                                                                                                                                                                                       (AFL. automation. shared. Data Labeler Widget. Ordinal Encoder
                                    property), 720
infrequent_categories_
                                                                                                                                                                                                                                                                       method), 719
                                    (AFL.automation.shared.DiffractionLabeler.OrdinianEersdertransform()
                                                                                                                                                                                                                                                                       (AFL. automation. shared. Diffraction Labeler. Ordinal Encoder\\
                                    attribute), 736
infrequent_categories_
                                                                                                                                                                                                                                                                       method), 739
                                     (AFL.automation.shared.DiffractionLabeler.OrdinapEncochersses_by_version()
                                    property), 740
                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.APIServer.ServiceInfo
init() (AFL.automation.APIServer.APIServer
                                                                                                                                                                                                                                                                      method), 143
                                     method), 17, 93, 150
                                                                                                                                                                                                                                 ip_addresses_by_version()
\verb"init_app"()" (AFL. automation. APIS erver. APIS erver. CORS")
                                                                                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                                     method), 95
                                                                                                                                                                                                                                                                       method), 755
init_app() (AFL.automation.APIServer.APIServer.JWTMapa@ddresses_by_version()
                                     method), 129
                                                                                                                                                                                                                                                                       (AFL.automation.shared.ServerDiscovery.ServiceInfo
init_buttons() (AFL.automation.shared.DatasetWidget.DatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetDatasetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidg
                                                                                                                                                                                                                                 IPVersion (class in AFL.automation.APIServer.APIServer),
                                    method), 74, 728, 732
init_casting_manifest()
                                      (AFL. automation. sample. Casting Server. Casting Se {\tt PWersion} (class in AFL. automation. shared. Server Discovery), and the sample of th
```

760	81, 782, 783
$\verb is_absolute() (AFL. automation. instrument. Seabreeze UVV \\ \verb sigma \\$	s_RoNarity() (in module
method), 202	AFL.automation.prepare.factory), 688
$\verb is_alive() (AFL. automation. APIS erver. APIS erver. Queue \textit{Distance}) \\$	
method), 135	AFL.automation.shared.units), 81, 783
is_alive() (AFL.automation.APIServer.QueueDaemon.Qies method), 181	ualdownth(i) (AFL.automation.instrument.SeabreezeUVVis.Path method), 201
	aenedra Reduct Dan (roth. automation.instrument. Seabreeze UVV is. Path
method), 795	method), 202 ns oeBoldingT(ly(AA F L.automation.instrument.SeabreezeUVVis.Path
method), 243	method), 202
is_alive() (AFL.automation.loading.LoadStopperDriver.Sts	
method), 246	AFL.automation.APIServer.QueueDaemon),
is_alive() (AFL.automation.loading.LoadStopperDriver.Sto	· -
· · · · · · · · · · · · · · · · · · ·	_serialized() (in module
is_alive()(AFL.automation.loading.Sensor.DummySensor1	· · · · · · · · · · · · · · · · · · ·
method), 323	781
is_alive() (AFL.automation.loading.Sensor.DummySensoirs method), 326	2_server_live() (AFL.automation.APIServer.APIServer.APIServer
	method), 16, 92, 149 Scors ReCANKAKE Lanctochation.instrument.Seabreeze UVV is.Path
method), 331	method), 202
is_alive() (AFL.automation.loading.SensorCallbackThreius	
method), 333	property), 59, 382, 384
is_alive() (AFL.automation.loading.SensorCallbackThreius	
method), 336	375
$\verb is_alive() (AFL. automation. loading. Sensor Callback Threius) \\$	
method), 339	379
is_alive() (AFL.automation.loading.SensorPollingThread.\$	
method), 345 is_alive() (AFL.automation.shared.ServerDiscovery.Run Tb	property), 59, 382, 384
method), 762	375
is_alive() (AFL.automation.shared.ServerDiscovery.Servis	
method), 767	erty), 379
	Usyma Paka () (AFL:automation.instrument.SeabreezeUVVis.Path
method), 202 is_char_device() (AFL.automation.instrument.Seabreezeils	method), 202 VVoid Hask() (in module
method), 202	Yvôd ftmt/() (in module AFL.automation.prepare.factory), 688
	_volume() (in module AFL.automation.shared.units),
AFL.automation.prepare.factory), 688	80, 783
	Daemon() (AFL.automation.APIServer.APIServer.QueueDaemon
AFL.automation.shared.units), 81, 782,	method), 135
	Daemon() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
is_density() (in module	method), 181
	Daemon() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.
783	method), 795
	Daemon() (AFL.automation.loading.LoadStopperDriver.SensorPollingT
method), 201	method), 243
method), 202	Daemon() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1 method), 246
$\verb is_file() (AFL. automation. instrument. Seabreeze UVV is. Pixton and the state of the state$	${\tt Daemon()} \ (AFL. automation. loading. Load Stopper Driver. Stop Load CBv2$
method), 201	method), 249
is_mass() (in module AFL.automation.prepare.factory), is	
688	method), 323
is mass() (in module AFL automation shared units) is	Daemon() (AFL automation loading Sensor DummySensor?

method), 325	jinja_loader (AFL.automation.APIServer.APIServer.Flask
isDaemon() (AFL.automation.loading.SensorCallbackThr	1 1 1
method), 331	jinja_options (AFL.automation.APIServer.APIServer.Flask
isDaemon() (AFL.automation.loading.SensorCallbackThr	
method), 333	join() (AFL.automation.APIServer.APIServer.QueueDaemon
isDaemon() (AFL.automation.loading.SensorCallbackThr	
method), 336	join() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
isDaemon() (AFL.automation.loading.SensorCallbackThr	•
method), 339	join() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDae
isDaemon() (AFL.automation.loading.SensorPollingThred	
method), 345	join() (AFL.automation.loading.LoadStopperDriver.SensorPollingThread
isDaemon() (AFL.automation.shared.ServerDiscovery.Run	
method), 762 isDaemon() (AFL.automation.shared.ServerDiscovery.Ser	join() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
method), 767	join() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
items() (AFL.automation.APIServer.Driver.PersistentCon	
method), 161	join() (AFL.automation.loading.Sensor.DummySensor1
items() (AFL.automation.APIServer.QueueDaemon.Data	
method), 179	join() (AFL.automation.loading.Sensor.DummySensor2
items() (AFL.automation.loading.OneSelectorBlowoutSat	
method), 268	ripieCeti.u ejjumitug;
items() (AFL.automation.loading.PneumaticPressureSam	
method), 278	join() (AFL.automation.loading.SensorCallbackThread.SimpleThreshold
items() (AFL.automation.loading.PneumaticSampleCell.c	
method), 288	join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
items() (AFL.automation.loading.PushPullSelectorSample	
method), 303	join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
items()(AFL.automation.loading.RSoXSSolutionSampleC	
method), 314	join() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
items()(AFL.automation.loading.TwoSelectorBlowoutSan	
method), 359	join() (AFL.automation.shared.ServerDiscovery.RunThread
items()(AFL.automation.shared.DatasetWidget.defaultdi	ct method), 762
method), 729	<pre>join() (AFL.automation.shared.ServerDiscovery.ServiceBrowser</pre>
items() (AFL.automation.shared.PersistentConfig.Mutable)	eMapping method), 767
method), 744	joinpath() (AFL. automation. instrument. Seabreeze UVV is. Path
$\verb items() (AFL. automation. shared. Persistent Config. Persistent C$	entConfig method), 202
method), 747	json (AFL.automation.APIServer.APIServer.Flask
<pre>iter_blueprints() (AFL.automation.APIServer.APISer</pre>	ver.Flask attribute), 106
method), 111	json_decoder(AFL.automation.APIServer.APIServer.Flask
<pre>iterate_protocols() (AFL.automation.prepare.Deck</pre>	property), 104
method), 370	json_encoder (AFL.automation.APIServer.APIServer.Flask
$\verb iterationid() (AFL. automation. APIS erver. APIS erver. M$	
method), 133	json_provider_class
iterationid() (AFL.automation.shared.MutableQueue.M	·- ·
method), 76, 742, 743	attribute), 105
iterdir() (AFL.automation.instrument.SeabreezeUVVis.I	
method), 200	AFL.automation.APIServer.APIServer), 85
1	justify_content(AFL.automation.prepare.DeckBuilderWidget.Layout
J	attribute), 432
jinja_env (AFL.automation.APIServer.APIServer.Flask	justify_content(AFL.automation.prepare.PrepareWidget.Layout
property), 107	attribute), 517
	winstify_content (AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 599
attribute), 103	justify_content(AFL.automation.prepare.SweepBuilderWidget.Layout
	Justing Contests (Ar L. amomanon. prepare. Sweep bunder wiaget. Layout

attribute), 663	keys (AFL.automation.prepare.PrepareWidget.VBox at-
$\verb"justify_items" (AFL. automation. prepare. Deck Builder Williams") and the property of the $	dget.Layoutribute), 541
attribute), 432	keys (AFL.automation.prepare.SampleSeriesWidget.Button
justify_items(AFL.automation.prepare.PrepareWidget.	
attribute), 517	keys (AFL.automation.prepare.SampleSeriesWidget.Checkbox
justify_items(AFL.automation.prepare.SampleSeriesW	
attribute), 599	keys (AFL.automation.prepare.SampleSeriesWidget.FloatText
justify_items(AFL.automation.prepare.SweepBuilderW	
attribute), 663	keys (AFL.automation.prepare.SampleSeriesWidget.HBox
jwt_required() (in module	attribute), 576
AFL.automation.APIServer.APIServer), 86	
	keys (AFL.automation.prepare.SampleSeriesWidget.IntText
JWTManager (class in AFL.automation.APIServer.APIServe	
128	keys (AFL.automation.prepare.SampleSeriesWidget.Label
K	attribute), 592
	${\tt keys} (AFL. automation. prepare. Sample Series Widget. Layout$
key (AFL.automation.APIServer.APIServer.ServiceInfo	attribute), 602
attribute), 141	keys (AFL.automation.prepare.SampleSeriesWidget.Text
key (AFL.automation.instrument.SeabreezeUVVis.Eq at-	attribute), 612
tribute), 197	${\bf keys}(AFL. automation. prepare. Sample Series Widget. VBox$
key (AFL.automation.shared.ServerDiscovery.AsyncServic	eInfo attribute), 620
attribute), 753	keys (AFL.automation.prepare.SweepBuilderWidget.Button
key (AFL.automation.shared.ServerDiscovery.ServiceInfo	attribute), 633
attribute), 770	keys (AFL.automation.prepare.SweepBuilderWidget.Checkbox
keys (AFL.automation.prepare.DeckBuilderWidget.Button	attribute), 641
attribute), 389	keys (AFL.automation.prepare.SweepBuilderWidget.HBox
keys (AFL.automation.prepare.DeckBuilderWidget.Checkl	
attribute), 397	keys (AFL.automation.prepare.SweepBuilderWidget.Label
keys (AFL.automation.prepare.DeckBuilderWidget.Dropde	
attribute), 410	keys (AFL.automation.prepare.SweepBuilderWidget.Layout
	attribute), 666
keys (AFL.automation.prepare.DeckBuilderWidget.HBox	keys (AFL.automation.prepare.SweepBuilderWidget.Text
attribute), 418	
keys (AFL.automation.prepare.DeckBuilderWidget.Label	attribute), 675
attribute), 425	keys (AFL.automation.prepare.SweepBuilderWidget.VBox
keys (AFL.automation.prepare.DeckBuilderWidget.Layout	
attribute), 435	keys() (AFL.automation.APIServer.Driver.PersistentConfig
keys (AFL.automation.prepare.DeckBuilderWidget.Text	method), 161
attribute), 443	keys() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
${\tt keys} \ (AFL. automation. prepare. Deck Builder Widget. VBox$	method), 179
attribute), 451	$\textbf{keys()} \ (AFL. automation. loading. One Selector Blowout Sample Cell. default to the property of the prop$
${\tt keys} (AFL. automation. prepare. Prepare Widget. Button at-$	method), 268
tribute), 478	${\tt keys()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. default defa$
keys (AFL.automation.prepare.PrepareWidget.Checkbox	method), 278
attribute), 486	keys() (AFL.automation.loading.PneumaticSampleCell.defaultdict
keys (AFL.automation.prepare.PrepareWidget.Dropdown	method), 288
attribute), 495	keys() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdic
keys (AFL.automation.prepare.PrepareWidget.HBox at-	method), 303
tribute), 503	${\tt keys()} \ (AFL. automation. loading. RSoXS Solution Sample Cell. default dict$
keys (AFL.automation.prepare.PrepareWidget.Label at-	method), 314
tribute), 510	keys() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.default
keys (AFL.automation.prepare.PrepareWidget.Layout	method), 359
attribute), 520	keys() (AFL.automation.shared.DatasetWidget.defaultdict
	method), 729
keys (AFL.automation.prepare.PrepareWidget.Text at-	keys() (AFL.automation.shared.PersistentConfig.MutableMapping
tribute), 533	Key 5 () (111 L. amonanon. sharea. 1 ersistem Conjig. Manaotemapping

method), 744

keys()(AFL. automation. shared. Persistent Config. Persi	<i>it</i> Caoyofigt	(AFL. automation. prepare. Sample Series Widget. Float Text)
	method), 747		attribute), 569
		layout	(AFL.automation.prepare.SampleSeriesWidget.HBox
L			attribute), 576
label(A	FL.automation.prepare.DeckBuilderWidget.Dropo	_l ∂ _æ yout	(AFL.automation.prepare.SampleSeriesWidget.IntText
	attribute), 410		attribute), 584
label(A	FL.automation.prepare.PrepareWidget.Dropdown	layout	(AFL.automation.prepare.SampleSeriesWidget.Label
	attribute), 495		attribute), 592
Label(c	lass in AFL.automation.prepare.DeckBuilderWidge	e p ayout	(AFL.automation.prepare.SampleSeriesWidget.Text
`	422	*	attribute), 612
Label(c	lass in AFL.automation.prepare.PrepareWidget),	layout	(AFL.automation.prepare.SampleSeriesWidget.VBox
	507		attribute), 620
Label(c	lass in AFL.automation.prepare.SampleSeriesWidg	_{zel} ąyout	(AFL.automation.prepare.SweepBuilderWidget.Button
	588		attribute), 633
Label(c	lass in AFL.automation.prepare.SweepBuilderWid	glayout	(AFL. automation. prepare. Sweep Builder Widget. Checkbox
	652		attribute), 641
label()	(AFL.automation.shared.DataLabelerWidget.Data	a Łabeie t	Afflequitomation.prepare.SweepBuilderWidget.HBox
	method), 71, 715, 721		attribute), 648
label()	(AFL. automation. shared. Diffraction Labeler. Di	a ะสงคะ สป	AFL.automation.prepare.SweepBuilderWidget.Label
	method), 75, 733, 741		attribute), 656
latch_s	shaker() (AFL.automation.prepare.Dummy_OT2_	_D7X9XFd_	Affilyautyppatjonyprepare.SweepBuilderWidget.Text
	method), 61, 460, 462		attribute), 676
layout(AFL.automation.prepare.DeckBuilderWidget.Butt	o_{h}^{1} ayout	(AFL.automation.prepare.SweepBuilderWidget.VBox
	attribute), 389		attribute), 683
layout(AFL.automation.prepare.DeckBuilderWidget.Chec	c <i>le</i> yyout	(class in AFL.automation.prepare.DeckBuilderWidget),
	attribute), 397		429
layout(AFL.automation.prepare.DeckBuilderWidget.Drop	hayqut	(class in AFL.automation.prepare.PrepareWidget),
	attribute), 410		514
layout(AFL. automation. prepare. Deck Builder Widget. HBo	χ Layout	(class in AFL.automation.prepare.SampleSeriesWidget),
	attribute), 418		596
layout(AFL. automation. prepare. Deck Builder Widget. Laberation and the properties of th	Layout	(class in AFL.automation.prepare.SweepBuilderWidget),
	attribute), 426		660
layout(AFL. automation. prepare. Deck Builder Widget. Text	1chmod	() (AFL.automation.instrument.SeabreezeUVVis.Path
	attribute), 443		method), 201
layout((AFL. automation. prepare. Deck Builder Widget. VBo.	χ left(A	FL.automation.prepare.DeckBuilderWidget.Layout
	attribute), 451		attribute), 432
layout	(AFL.automation.prepare.PrepareWidget.Button	left	(AFL.automation.prepare.PrepareWidget.Layout
	attribute), 478		attribute), 517
layout(AFL. automation. prepare. Prepare Widget. Checkbox and the contraction of the contracti	χ left(A	FL.automation.prepare.SampleSeriesWidget.Layout
	attribute), 486		attribute), 599
layout(AFL. automation. prepare. Prepare Widget. Dropdown	η_{t} left (A	FL.automation.prepare.SweepBuilderWidget.Layout
	attribute), 495		attribute), 663
layout	(AFL.automation.prepare.PrepareWidget.HBox	link_to	o() (AFL.automation.instrument.SeabreezeUVVis.Path
	attribute), 503		method), 201
layout	(AFL. automation. prepare. Prepare Widget. Label	listen	ers (AFL.automation.APIServer.APIServer.Zeroconf
	attribute), 511		property), 146
layout(AFL.automation.prepare.PrepareWidget.Text at-	listen	ers (AFL.automation.shared.ServerDiscovery.Zeroconf
	tribute), 533		property), 776
layout	(AFL.automation.prepare.PrepareWidget.VBox	listify	(in module

attribute), 541

attribute), 550

attribute), 558

 ${\tt layout} (AFL. automation. prepare. Sample Series Widget. Buttle is {\tt stify()}$

 ${\tt layout} (AFL. automation. prepare. Sample Series Widget. Check \verb|isk| if y ()$

AFL.automation.APIServer.APIServer), 86

(in AFL.automation.APIServer.Driver), 157

(in AFL.automation.APIServer.DummyDriver),

module

module

163			method), 266
listify()	(in	module 1	${\tt oadSample()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pressure $
AFL.autor	mation.APIServer.Dummy	OT2Driver),	method), 40, 275, 279
170		1	${\tt oadSample()}\ (AFL. automation. loading. Pneumatic Pressure Sample Cell. S$
listify()	(in	module	method), 277
AFL.autor 207	nation.instrument.SpecScr	reen_Driver), 1	oadSample() (AFL.automation.loading.PneumaticSampleCell.Pneumatic method), 41, 285, 289
listify() (in mod 689	lule AFL.automation.prepa	are.factory), 1	oadSample() (AFL.automation.loading.PneumaticSampleCell.SampleCe method), 287
listify()	(in mation.sample.CastingSer		oadSample() (AFL.automation.loading.PushPullSelectorSampleCell.Pus method), 44, 299, 304
693	name maamp ver e aasum 8 ser		oadSample() (AFL.automation.loading.PushPullSelectorSampleCell.Sam
listify() (in mod	lule AFL.automation.share		method), 301
81, 783, 7			${\tt oadSample()}\ (AFL. automation. loading. RSoXSS olutionSampleCell. RSoX$
			ckBuildemWildkger), 46, 310, 316
	60, 404, 455	-	oadSample() (AFL.automation.loading.RSoXSSolutionSampleCell.Samp
* * * * * * * * * * * * * * * * * * * *	utomation.prepare.Prepare		
method),		~	oadSample() (AFL.automation.loading.SampleCell.SampleCell
* * * * * * * * * * * * * * * * * * * *	utomation.prepare.Prepare		
method),		-	oadSample() (AFL.automation.loading.TwoSelectorBlowoutSampleCell
	utomation.prepare.StockBu		
	66, 626, 627		oadSample()(AFL.automation.loading.TwoSelectorBlowoutSampleCell.
			r.ServiceInfohod), 55, 356, 361
method),	143	L	oadStopperDriver (class in
load_from_cache	() (AFL.automation.share		ery.Asyn ASE Exi ueon fation.loading.LoadStopperDriver),
method), '			36, 237, 250
		d.ServerDiscol	oc)(AdriviauInfnation.prepare.Component.ParseException
method), '			attribute), 383
* * * * * * * * * * * * * * * * * * * *		ADLiveProcess	oGl(ArRICdinenomation.prepare.DeckBuilderWidget.Button
	24, 792, 793		attribute), 389
		re.Dummy_O T	<mark>ддЮл</mark> Герина и под правения и при при при при при при при при при п
	62, 460, 462	•	attribute), 397
		re.OT2Client. Y	AGCA Fantutomation.prepare.DeckBuilderWidget.Dropdown
	62, 468, 470		attribute), 410
<pre>load_instrument method),</pre>		le.CastingServ l e	TACTienutomation.prepare.DeckBuilderWidget.HBox attribute), 418
load_labware()(AFL.automation.EpicsADI	LiveProcess.C l i	ogt. CAiFhtautomation.prepare.DeckBuilderWidget.Label
	24, 792, 793		attribute), 426
		Dummy_ <i>OT2</i> _ I D	ogye A Dlummiyn @Ti&n Drega re.DeckBuilderWidget.Layout
	61, 460, 462	· — —	attribute), 435
<pre>load_labware()(</pre>		OT2Client.OT 2 0	6kj ent(AFL.automation.prepare.DeckBuilderWidget.Text attribute), 444
<pre>load_labware() (method), '</pre>	-	astingServer. Q I	EXCMAR L.automation.prepare.DeckBuilderWidget.VBox attribute), 451
LoaderCommunica	tion (class	in 1	og (AFL.automation.prepare.PrepareWidget.Button at-
AFL.autor	mation.loading.SensorCall		tribute), 478
52, 328, 3			og (AFL.automation.prepare.PrepareWidget.Checkbox
<pre>loadSample() (AF</pre>	L.automation.APIServer.L		
method),	21, 168, 170	1	og (AFL.automation.prepare.PrepareWidget.Dropdown
	L.automation.APIServer.L		
method),	22, 174, 176	1	og (AFL.automation.prepare.PrepareWidget.HBox at-
			utSample@dlu@neSelectorBlowoutSampleCell
method),			og (AFL.automation.prepare.PrepareWidget.Label at-
<pre>loadSample() (AF</pre>	L.automation.loading.One	eSelectorBlowoi	utSample CidutE);ø\$electorBlowoutSampleCell

Log	(AFL.automation.prepare.PrepareWidget.Layout attribute), 520	log_exc	eption_warning() (AFL.automation.APIServer.APIServer.Zeroconf
Log	(AFL.automation.prepare.PrepareWidget.Text	_	class method), 149
_	attribute), 534	log_exc	eption_warning()
Log	(AFL.automation.prepare.PrepareWidget.VBox at-		(AFL.automation.shared.ServerDiscovery.Zeroconf
	tribute), 541		class method), 779
Log((AFL.automation.prepare.SampleSeriesWidget.Button attribute), 550	log_war	ning_once() (AFL.automation.APIServer.APIServer.Zeroconf class method), 149
Log(okog_war	<pre>ning_once() (AFL.automation.shared.ServerDiscovery.Zeroconf</pre>
Log((AFL.automation.prepare.SampleSeriesWidget.FloatTe. attribute), 569	xlrogged_	in() (AFL.automation.APIServer.Client.Client method), 18, 152, 155
Log	(AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 576	logged_	in() (AFL.automation.EpicsADLiveProcess.Client.Client method), 24, 792, 793
Log((AFL.automation.prepare.SampleSeriesWidget.IntText attribute), 584	logged_	in() (AFL.automation.loading.LoadStopperDriver.Client method), 234
Log		logged_	in() (AFL.automation.prepare.DeckBuilderWidget.Client method), 402
Log((AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 602	logged_	
Log	(AFL.automation.prepare.SampleSeriesWidget.Text attribute), 613	logged_	in() (AFL.automation.prepare.OT2Client.OT2Client method), 468
Log	(AFL.automation.prepare.SampleSeriesWidget.VBox attribute), 620	logged_	in() (AFL.automation.prepare.SampleSeriesWidget.Client method), 563
Log((AFL.automation.prepare.SweepBuilderWidget.Button attribute), 633	logged_	in() (AFL.automation.sample.CastingServer.Client method), 698
Log((AFL.automation.prepare.SweepBuilderWidget.Checkb attribute), 641	∂lxogged_	in() (AFL.automation.sample.CastingServer.OT2Client method), 704
Log((AFL.automation.prepare.SweepBuilderWidget.HBox attribute), 648	logger	(AFL.automation.APIServer.APIServer.Flask property), 107
Log((AFL.automation.prepare.SweepBuilderWidget.Label attribute), 656	LoggerF	
Log((AFL.automation.prepare.SweepBuilderWidget.Layout	LoggerF	
	attribute), 666		AFL.automation.APIServer.LoggerFilter),
Log	(AFL.automation.prepare.SweepBuilderWidget.Text		22, 176
	attribute), 676	login()	(AFL.automation.APIServer.APIServer.APIServer
Log	(AFL.automation.prepare.SweepBuilderWidget.VBox		method), 17, 93, 150
	attribute), 683	login()	(AFL.automation.APIServer.Client.Client
Log_	_event() (AFL.automation.sample.CastingServer.Cast	tingServer	method), 18, 152, 155
	method), 82, 695, 705	-	(AFL.automation.EpicsADLiveProcess.Client.Client
Log_	_exception() (AFL.automation.APIServer.APIServer.		method), 24, 792, 793
J	method), 115		(AFL.automation.loading.LoadStopperDriver.Client
Log_	_exception_debug()	5 0	method), 234
J-		login()	(AFL.automation.prepare.DeckBuilderWidget.Client
	class method), 149	5 0	method), 403
Loa	_exception_debug()	login()	
- 5-	(AFL.automation.shared.ServerDiscovery.Zeroco	_	method), 464
	class method), 778	•	(AFL.automation.prepare.OT2Client.OT2Client
Loa	_exception_once()	- 5()	method), 468
- 5-		login()	(AFL.automation.prepare.SampleSeriesWidget.Client
	class method), 149	9-11()	method), 564
Loa	_exception_once()	login()	(AFL.automation.sample.CastingServer.Client
- 5-	(AFL.automation.shared.ServerDiscovery.Zeroco		method), 698
	class method), 779		(AFL.automation.sample.CastingServer.OT2Client

```
method), 704
                                                                                                                                                                                                                                                                                  (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), 66, 608, 625
                                                                                                                                                                                                                                          make_pipette_params()
                                      method), 17, 93, 151
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), 201
                                                                                                                                                                                                                                                                                  method), 66, 608, 624
                                                                                                                                                                                                                                          make_protocol()
                                                                                                                                                                                                                                                                                                                                                (AFL.automation.prepare.Deck
M
                                                                                                                                                                                                                                                                                  method), 370
\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), 526
                                       method), 108
\verb|make_align_script()| (AFL. automation. prepare. Deck | \verb|make_protocol()| (AFL. automation. prepare. Sample Series Widget. Sampl
                                                                                                                                                                                                                                                                                  method), 65, 606, 624
                                       method), 370
make_all_labels() (AFL.automation.prepare.PrepareWilledisScienspessesWildEL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                  method), 118
                                       method), 527
make_all_labels() (AFL.automation.prepare.SampleSeries Widget
                                                                                                                                                                                                                                                                                  (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                                                                                                                                                                         method),
                                       method), 65, 607, 624
                                                                                                                                                                                                                                                                                  370
make_all_labels_cb()
                                       (AFL.automation.prepare.PrepareWidget.SampleSerkesWAGEipt()
                                                                                                                                                                                                                                                                                                                                                (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                                                                  method), 370
                                      method), 527
                                                                                                                                                                                                                                          make_shell_context()
make_all_labels_cb()
                                      (AFL. automation. prepare. Sample Series Widget. Sample Series Widget automation. API Server. API Server. Flask automation. API Server. 
                                                                                                                                                                                                                                                                                  method), 109
                                      method), 65, 607, 624
make_binary_plot()(AFL.automation.prepare.SweepBuilleteVisiteOf.SweepBuilletevwuttoptanippwprepare.PrepareWidget.StockBuilderV
                                                                                                                                                                                                                                                                                  method), 528
                                      method), 68, 671, 687
                                                                                                                                                                                                                                          make_stock_cb() (AFL.automation.prepare.StockBuilderWidget.StockBuil
make_catch_protocol()
                                       (AFL.automation.prepare.PrepareWidget.SampleSeriesWidgethod), 66, 626, 628
                                                                                                                                                                                                                                          make_stock_grid() (AFL.automation.prepare.SweepBuilderWidget.Swee
                                      method), 526
                                                                                                                                                                                                                                                                                  method), 68, 671, 687
make_catch_protocol()
                                       (AFL. automation. prepare. Sample Series Widget. Sample Series Widget. Stock Builder W
                                                                                                                                                                                                                                                                                 method), 67, 627, 628
                                      method), 65, 606, 624
                                                                                                                                                                                                                                          make_target_component_masses()
make_component_grid()
                                       (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget_Equtypeqtion.prepare.MassBalance
                                                                                                                                                                                                                                                                                  method), 372
                                      method), 66, 608, 624
make_config() (AFL.automation.APIServer.APIServer.Flamake_ternary_plot()
                                                                                                                                                                                                                                                                                  (AFL.automation.prepare.SweepBuilderWidget.SweepBuilderWid
                                      method), 108
                                                                                                                                                                                                                                                                                  method), 68, 671, 687
make_default_options_response()
                                                                                                                                                                                                                                          make_wellplate_locs()
                                                                                                                                                                                                                                                                                                                                                                                                                                              module
                                                                                                                                                                                                                                                                                                                                                                                             (in
                                       (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                 AFL.automation.prepare), 367
                                      method), 116
make_grid_mask() (AFL.automation.prepare.MassBalanc@make_wellplate_locs()
                                                                                                                                                                                                                                                                                                                                                                                                                                              module
                                                                                                                                                                                                                                                                                                                                                                                             (in
                                                                                                                                                                                                                                                                                  AFL.automation.prepare.utilities), 69, 692
                                      method), 372
                                                                                                                                                                                                                                          makeRegistar()
                                                                                                                                                                                                                                                                                                                                                                                                                                              module
make_locs() (in module AFL.automation.prepare), 367
                                                                                                                                                                                                                                                                                 AFL.automation.prepare.PrepType), 470
make_locs()
                                                                                                                                                                                                    module
                                                                                                                                                                                                                                          makeRegistar()
                                                                                                                                                                                                                                                                                                                                                                                                                                              module
                                                                                                                                                                                                                                                                                                                                                                           (in
                                      AFL.automation.prepare.utilities), 69, 692
                                                                                                                                                                                                                                                                                 AFL.automation.shared.utilities), 81, 784
make_mass_balance() (AFL.automation.prepare.Deck
                                                                                                                                                                                                                                          makeRegistrar()
                                                                                                                                                                                                                                                                                                                                                                                                                                              module
                                                                                                                                                                                                                                                                                                                                                                              (in
                                       method), 370
                                                                                                                                                                                                                                                                                 AFL.automation.APIServer.Driver),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                19.
make_mass_fraction_matrix()
                                                                                                                                                                                                                                                                                  157, 162
                                       (AFL.automation.prepare.MassBalance
                                                                                                                                                                                                                                          manage_service_info_to_list()
                                      method), 371
                                                                                                                                                                                                                                                                                  (AFL.automation.APIServer.Client.ServerDiscovery
make_mixing_wells()
                                       (AFL. automation. prepare. Prepare Widget. Sample Series Widget thod),\ 154
                                                                                                                                                                                                                                          manage_service_info_to_list()
                                      method), 527
                                                                                                                                                                                                                                                                                  (AFL.automation.shared.ServerDiscovery.ServerDiscovery
make_mixing_wells()
                                       (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWtlage), 78, 764, 779
                                                                                                                                                                                                                                          margin (AFL.automation.prepare.DeckBuilderWidget.Layout
                                      method), 65, 607, 624
                                                                                                                                                                                                                                                                                  attribute), 432
make_mixing_wells()
```

```
margin (AFL.automation.prepare.PrepareWidget.Layout max_width(AFL.automation.prepare.SweepBuilderWidget.Layout
         attribute), 517
                                                                attribute), 663
margin (AFL.automation.prepare.SampleSeriesWidget.Layon@asure_out() (AFL.automation.prepare.factory.Solution
         attribute), 599
                                                                method), 691
margin(AFL.automation.prepare.SweepBuilderWidget.Laymerasure_out()
                                                                           (AFL.automation.prepare.Solution
        attribute), 663
                                                                method), 378
Markdown (class in AFL.automation.shared.widgetui), metadata (AFL.automation.shared.widgetui.Markdown
                                                                attribute), 788
mask_serialized_objs()
                                                      min_height (AFL.automation.prepare.DeckBuilderWidget.Layout
        (AFL.automation.APIServer.APIServer.QueueDaemon
                                                                attribute), 433
        method), 136
                                                      min_height(AFL.automation.prepare.PrepareWidget.Layout
mask_serialized_objs()
                                                                attribute), 518
         (AFL.automation.APIServer.QueueDaemon.QueumDnehweinght (AFL.automation.prepare.SampleSeriesWidget.Layout
        method), 23, 181, 182
                                                                attribute), 599
mass (AFL.automation.prepare.Component.Component
                                                      min\_height (AFL. automation. prepare. Sweep Builder Widget. Layout
         property), 59, 381, 384
                                                                attribute), 663
mass (AFL.automation.prepare.factory.Solution prop-
                                                      min_width(AFL.automation.prepare.DeckBuilderWidget.Layout
         erty), 690
                                                                attribute), 433
mass (AFL.automation.prepare.Solute property), 375
                                                      min_width(AFL.automation.prepare.PrepareWidget.Layout
mass (AFL. automation. prepare. Solution property), 377
                                                                attribute), 518
                                                      \verb|min_width| (AFL. automation. prepare. Sample Series Widget. Layout
mass (AFL.automation.prepare.Solvent property), 379
mass_fraction(AFL.automation.prepare.factory.Solution
                                                                attribute), 599
        property), 691
                                                      min\_width (AFL.automation.prepare.SweepBuilderWidget.Layout
mass_fraction
                     (AFL.automation.prepare.Solution
                                                                attribute), 663
        property), 378
                                                      MixingException, 79, 780, 781
mass_totals_component()
                                                      mkdir() (AFL.automation.instrument.SeabreezeUVVis.Path
         (AFL.automation.prepare.SampleSeries
                                                                method), 200
        method), 374
                                                      model\_id(AFL.automation.prepare.DeckBuilderWidget.Button
mass_totals_stock()
                                                               property), 389
         (AFL.automation.prepare.SampleSeries
                                                      model_id(AFL.automation.prepare.DeckBuilderWidget.Checkbox
         method), 374
                                                                property), 397
MassBalance (class in AFL.automation.prepare), 370
                                                      model_id(AFL.automation.prepare.DeckBuilderWidget.Dropdown
match() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                               property), 410
         method), 202
                                                      model\_id(AFL.automation.prepare.DeckBuilderWidget.HBox)
match_server_by_name()
                                                                property), 418
         (AFL.automation.APIServer.Client.ServerDiscovermodel_id(AFL.automation.prepare.DeckBuilderWidget.Label
        method), 155
                                                               property), 426
match_server_by_name()
                                                      model\_id(AFL.automation.prepare.DeckBuilderWidget.Layout
         (AFL.automation.shared.ServerDiscovery.ServerDiscovery property), 436
        method), 79, 764, 780
                                                      model\_id(AFL.automation.prepare.DeckBuilderWidget.Text
max_height (AFL.automation.prepare.DeckBuilderWidget.Layout property), 444
         attribute), 432
                                                      model_id(AFL.automation.prepare.DeckBuilderWidget.VBox
max_height(AFL.automation.prepare.PrepareWidget.Layout
                                                                property), 451
         attribute), 517
                                                      model\_id(AFL.automation.prepare.PrepareWidget.Button
max_height (AFL.automation.prepare.SampleSeriesWidget.Layout property), 479
         attribute), 599
                                                      model\_id(AFL.automation.prepare.PrepareWidget.Checkbox)
max_height (AFL.automation.prepare.SweepBuilderWidget.Layout property), 486
                                                      model\_id(AFL.automation.prepare.PrepareWidget.Dropdown
         attribute), 663
max_width(AFL.automation.prepare.DeckBuilderWidget.Layout
                                                               property), 495
         attribute), 433
                                                      model_id(AFL.automation.prepare.PrepareWidget.HBox
{\tt max\_width} \ (AFL. automation. prepare. Prepare Widget. Layout
                                                                property), 503
                                                      model\_id(AFL.automation.prepare.PrepareWidget.Label
        attribute), 518
max_width(AFL.automation.prepare.SampleSeriesWidget.Layout
                                                               property), 511
```

model_id(AFL.automation.prepare.PrepareWidget.Layout

attribute), 599

property), 521	AFL.automation.EpicsADLiveProcess.Client,
model_id (AFL.automation.prepare.PrepareWidget.Text	24, 791 AFI automation EnjoyAPI ivoProcess ReducePageon
<pre>property), 534 model_id(AFL.automation.prepare.PrepareWidget.VBox</pre>	AFL.automation.EpicsADLiveProcess.ReduceDaemon, 24,793
property), 541	AFL.automation.instrument, 25, 183
model_id (AFL.automation.prepare.SampleSeriesWidget.Button	
property), 550	183
$\verb model_id (AFL. automation. prepare. Sample Series Widget. Check Check $	
property), 558	188
$\verb model_id (AFL. automation. prepare. Sample Series Widget. Float Total Control of the Contro$	
property), 569	AFL.automation.instrument.NetworkCamera,
$\verb model_id (AFL. automation. prepare. Sample Series Widget. HBox $	27, 193
property), 577	AFL.automation.instrument.SeabreezeUVVis,
model_id(AFL.automation.prepare.SampleSeriesWidget.IntTex	
property), 585	AFL.automation.instrument.SpecScreen_Driver,
<pre>model_id(AFL.automation.prepare.SampleSeriesWidget.Label</pre>	28, 207 AFL.automation.loading, 29, 213
model_id (AFL.automation.prepare.SampleSeriesWidget.Layou	——————————————————————————————————————
property), 602	31, 215
model_id(AFL.automation.prepare.SampleSeriesWidget.Text	
property), 613	31, 216
	AFL.automation.loading.DigitalOutPressureController,
property), 620	33, 223
	nAFL.automation.loading.DoubleViciMultiposSelector,
property), 633	34, 225
model_id(AFL.automation.prepare.SweepBuilderWidget.Check	
property), 641	AFL.automation.loading.FlowSelector, 35,
$\verb model_id (AFL. automation. prepare. SweepBuilder Widget. HBox $	232
property), 648	AFL.automation.loading.LoadStopperDriver,
$\verb model_id (AFL. automation. prepare. Sweep Builder Widget. Label \\$	35, 232
property), 656	AFL.automation.loading.MultiChannelRelay,
$\verb model_id (AFL. automation. prepare. Sweep Builder Widget. Layout and the state of the state$	
property), 666	AFL.automation.loading.NE1kSyringePump,
model_id(AFL.automation.prepare.SweepBuilderWidget.Text	37, 252
property), 676	AFL.automation.loading.OneSelectorBlowoutSampleCell,
model_id(AFL.automation.prepare.SweepBuilderWidget.VBox	38, 256
property), 683	AFL.automation.loading.PneumaticPressureSampleCell,
module AFI sutemation 13 780	39, 270
AFL.automation, 13, 789 AFL.automation.APIServer, 14, 83	AFL.automation.loading.PneumaticSampleCell, 41,280
AFL.automation.APIServer.APIServer, 14, 83	AFL.automation.loading.PressureController,
AFL.automation.APIServer.Client, 17, 151	42, 289
AFL.automation.APIServer.Driver, 19, 156	AFL.automation.loading.PressureControllerAsPump,
AFL.automation.APIServer.DummyDriver, 20,	42, 291
163	AFL.automation.loading.PushPullSelectorSampleCell,
AFL.automation.APIServer.DummyOT2Driver,	43, 294
21, 170	AFL.automation.loading.RSoXSSolutionSampleCell,
AFL.automation.APIServer.LoggerFilter,	45, 305
22, 176	AFL.automation.loading.SainSmartRelay,
AFL.automation.APIServer.QueueDaemon, 22,	47, 316
176	AFL.automation.loading.SampleCell, 48, 320
AFL.automation.EpicsADLiveProcess, 23, 790	AFL.automation.loading.Sensor, 48, 321
AFL.automation.EpicsADLiveProcess.AreaDetector	_
23, 790	50, 328

AFL.automation.loading.SensorPollingThread 52,343	d, AFL.automation.shared.utilities, 81, 783 AFL.automation.shared.widgetui, 81, 784
	molarity (AFL.automation.prepare.factory.Solution
347	property), 691
AFL.automation.loading.SyringePump, 53, 348	molarity (AFL.automation.prepare.Solution property), 378
AFL.automation.loading.Tubing, 54, 349	moles (AFL.automation.prepare.Component.Component
AFL.automation.loading.TwoSelectorBlowout	
54, 350	moles (AFL.automation.prepare.Solute property), 375
AFL.automation.loading.UltimusVPressureCo	nttobes en FL. automation. prepare. Solvent property), 379
56, 361	move() (AFL.automation.APIServer.APIServer.MutableQueue
AFL.automation.loading.ViciMultiposSelector	or, method), 133
57, 364	move() (AFL.automation.shared.MutableQueue.MutableQueue
AFL.automation.prepare, 57, 367	method), 76, 743
AFL.automation.prepare.Component, 58, 380	<pre>move_item() (AFL.automation.APIServer.APIServer.APIServer</pre>
AFL.automation.prepare.DeckBuilderWidget,	method), 17, 93, 150
59, 384	<pre>move_item() (AFL.automation.APIServer.Client.Client</pre>
AFL.automation.prepare.Dummy_OT2_Driver,	method), 18, 153, 156
61, 455	<pre>move_item() (AFL.automation.loading.LoadStopperDriver.Client</pre>
AFL.automation.prepare.factory, 68, 687	method), 235
AFL.automation.prepare.OT2Client, 62, 463	<pre>move_item() (AFL.automation.prepare.DeckBuilderWidget.Client</pre>
AFL.automation.prepare.PrepareWidget, 63,	method), 403
473	<pre>move_item() (AFL.automation.prepare.OT2Client.Client</pre>
AFL.automation.prepare.PrepType, 63, 470	method), 465
AFL.automation.prepare.SampleSeriesWidget	move_item() (AFL.automation.prepare.OT2Client.OT2Client
64, 545	method), 468
AFL.automation.prepare.StockBuilderWidget	move_item() (AFL.automation.prepare.SampleSeriesWidget.Client
66, 625	method), 564
AFL.automation.prepare.SweepBuilderWidget	move_item() (AFL.automation.sample.CastingServer.Client
67, 628	method), 699
AFL.automation.prepare.utilities, 69, 692	<pre>move_item() (AFL.automation.sample.CastingServer.OT2Client</pre>
AFL.automation.sample, 82, 692	method), 704
AFL.automation.sample.CastingServer, 82,	$move_temp()$ (AFL.automation.sample_env.TemperatureDeck.Temperature
693	method), 69, 709, 710
AFL.automation.sample_env, 69, 705	<pre>mpl_plot_to_bytes()</pre>
AFL.automation.sample_env.TemperatureDeck	AFL.automation.shared.utilities), 81, 784
69, 705	msg (AFL.automation.prepare.Component.ParseException
AFL.automation.shared, 70, 710	attribute), 383
AFL.automation.shared.DataLabelerWidget,	MultiChannelRelay (class in
70, 711	AFL.automation.loading.MultiChannelRelay),
AFL.automation.shared.DatasetWidget, 72,	36, 251
722	MultiChannelRelay (class in
AFL.automation.shared.DiffractionLabeler, 74,732	AFL.automation.loading.SainSmartRelay), 317
AFL.automation.shared.exceptions, 79, 780	MutableMapping (class in
AFL.automation.shared.MutableQueue, 75,	AFL.automation.shared.PersistentConfig),
742	744
AFL.automation.shared.PersistentConfig,	MutableQueue (class in
76, 743	AFL.automation.APIServer.APIServer), 133
AFL.automation.shared.serialization, 80, 781	MutableQueue (class in AFL.automation.shared.MutableQueue),
AFL.automation.shared.ServerDiscovery, 77,748	76, 742, 743

 ${\tt AFL.automation.shared.units},\,80,\,782$

property), 181

N		property), 795		
n_features_in_(AFL.automation.shared.DataLabelerWiattribute), 716		ν		
n_features_in_(AFL.automation.shared.DiffractionLaberattribute), 736		property), 240		
name (AFL.automation.APIServer.APIServer.FileHandler property), 97		id (AFL.automation.loa property), 249		_
name (AFL.automation.APIServer.APIServer.Flask prop-		id(AFL.automation.loa property), 323		
name (AFL.automation.APIServer.APIServer.QueueDaemor property), 136		property), 320		
name (AFL.automation.APIServer.APIServer.ServiceInfo		id(AFL.automation.loa property), 331		
name (AFL.automation.APIServer.APIServer.SMTPHandler property), 138		property), 334		
name (AFL.automation.APIServer.QueueDaemon.QueueDa property), 181		property), 551		
name (AFL.automation.EpicsADLiveProcess.ReduceDaemo property), 795		property), 540		
nronerty) 202		id (AFL.automation.loa property), 346		· ·
name (AFL.automation.loading.LoadStopperDriver.SensorF property), 244		property), 702		
name (AFL.automation.loading.LoadStopperDriver.StopLoa property), 246		property), 108		
name (AFL.automation.loading.LoadStopperDriver.StopLoaproperty), 249		Ar L.uutomutton.touutn	(class g.NE1kSyringePump)	in ,
name (AFL.automation.loading.Sensor.DummySensor1		37,252,255 resh_token_loader()	1	
property), 323		(AFL.automation.APIS		Manager
name (AFL.automation.loading.Sensor.DummySensor2		<i>method</i>), 131	cr ver.211 15er ver.5 W 11	nanager
property), 326			(class	in
name (AFL.automation.loading.SensorCallbackThread.Sens		AFL.automation.instrui	`	
name (AFL.automation.loading.SensorCallbackThread.Simp property), 334 name (AFL.automation.loading.SensorCallbackThread.Stop	next_bu	tton_callback()	d.DataLahelerWidge	t.DataLahelerWidget
property), 337 name (AFL.automation.loading.SensorCallbackThread.Stop		memoa), /1, /14, /21		
property), 340		(Ar L.automation.snare	d.DatasetWidget.Date	asetWidget
name (AFL.automation.loading.SensorPollingThread.Senso property), 345	nex t_bu	cton_carrback()	15.00	D
name (AFL.automation.shared.ServerDiscovery.AsyncService attribute), 755	cernjo	(AFL.automation.share method), 75, 733, 741		DiffractionLabeler
name (AFL.automation.shared.ServerDiscovery.RunThread property), 762	Notround	JELLUL, 00, 700, 701		_
name (AFL.automation.shared.ServerDiscovery.ServiceBrov property), 767		metnoa), 140		
name (AFL.automation.shared.ServerDiscovery.ServiceInfo attribute), 772		methoa), //b		
native_id(AFL.automation.APIServer.APIServer.QueueL		metnoa), 590		
native_id(AFL.automation.APIServer.QueueDaemon.Qu	คณะ D ส์ย _m ย	shange() (AFL.automa method), 398	tion.prepare.DeckBu	lderWidget.Checkbo

860 Index

 $\verb|native_id| (AFL. automation. Epics ADLive Process. Reduce Daemon. Reduce Daem$

- method), 411 method), 666
- notify_change() (AFL.automation.prepare.DeckBuilderWidgitfyBahange() (AFL.automation.prepare.SweepBuilderWidget.Text method), 418 method), 676
- notify_change() (AFL.automation.prepare.DeckBuilderWidgitfyabhlange() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 426 method), 683
- notify_change() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 436
- notify_change() (AFL.automation.prepare.DeckBuilderWidget-TexFit (AFL.automation.prepare.DeckBuilderWidget.Layout method), 444 attribute), 433
- notify_change() (AFL.automation.prepare.DeckBuilderWidget\(\frac{VBP\}{2}\)t (AFL.automation.prepare.PrepareWidget.Layout method), 451 attribute), 518
- notify_change() (AFL.automation.prepare.PrepareWidgenBytten_fit (AFL.automation.prepare.SampleSeriesWidget.Layout method), 479 attribute), 600
- notify_change() (AFL.automation.prepare.PrepareWidgetGbett (AFL.automation.prepare.SweepBuilderWidget.Layout method), 487 attribute), 664
- notify_change() (AFL.automation.prepare.PrepareWidgetDropdowsition (AFL.automation.prepare.DeckBuilderWidget.Layout method), 496 attribute), 433
- notify_change() (AFL.automation.prepare.PrepareWidgetHeet_position (AFL.automation.prepare.PrepareWidget.Layout method), 503 attribute), 518
- notify_change() (AFL.automation.prepare.PrepareWidgetJztet_position(AFL.automation.prepare.SampleSeriesWidget.Layout method), 511

 attribute), 600
- notify_change() (AFL.automation.prepare.PrepareWidgetJzectt_position(AFL.automation.prepare.SweepBuilderWidget.Layout method), 521 attribute), 664
- notify_change() (AFL.automation.prepare.PrepareWidgetTeetve() (AFL.automation.prepare.DeckBuilderWidget.Button method), 534 method), 390
- notify_change() (AFL.automation.prepare.PrepareWidget\Serve() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 541 method), 398
- notify_change() (AFL.automation.prepare.SampleSeries \ighthigget\ButtaqAFL.automation.prepare.DeckBuilderWidget.Dropdown method), 551 method), 411
- notify_change() (AFL.automation.prepare.SampleSeries Widget Clock PFL.automation.prepare.DeckBuilder Widget.HBox method), 559

 method), 418
- notify_change() (AFL.automation.prepare.SampleSeries WidgetvElog(KAFL.automation.prepare.DeckBuilderWidget.Label method), 569 method), 426
- notify_change() (AFL.automation.prepare.SampleSeries Widget AFL.automation.prepare.DeckBuilderWidget.Layout method), 577 method), 436
- notify_change() (AFL.automation.prepare.SampleSeries Widget Let Text Lautomation.prepare.DeckBuilderWidget.Text method), 585 method), 444
- notify_change() (AFL.automation.prepare.SampleSeries \ightharpoonup \ightharpoonu
- notify_change() (AFL.automation.prepare.SampleSeries WidgetVeCy)(AFL.automation.prepare.PrepareWidget.Button method), 602 method), 479
- notify_change() (AFL.automation.prepare.SampleSeries \(\) (AFL.automation.prepare.PrepareWidget.Checkbox method), 613 \(method), 487 \)
- notify_change() (AFL.automation.prepare.SampleSeries WidgetvElby(AFL.automation.prepare.PrepareWidget.Dropdown method), 496
- notify_change() (AFL.automation.prepare.SweepBuilder\@get\Chyqk\pe_automation.prepare.PrepareWidget.Label method), 641 method), 511
- notify_change() (AFL.automation.prepare.SweepBuilder Widget-Lautomation.prepare.PrepareWidget.Text method), 656 method), 534
- $\verb"notify_change" () \textit{ (AFL. automation. prepare. Sweep Builder Widget. Layout a)} \\$

- observe() (AFL.automation.prepare.PrepareWidget.VBox on_msg() (AFL.automation.prepare.DeckBuilderWidget.Text method), 541 method), 444
- observe() (AFL.automation.prepare.SampleSeriesWidget.Bouttomsg() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 551 method), 452
- observe() (AFL.automation.prepare.SampleSeriesWidget.**Checkso**() (AFL.automation.prepare.PrepareWidget.Button method), 559 method), 479
- observe() (AFL.automation.prepare.SampleSeriesWidget.FokoantEgy() (AFL.automation.prepare.PrepareWidget.Checkbox method), 569 method), 487
- observe() (AFL.automation.prepare.SampleSeriesWidget.MBomsg() (AFL.automation.prepare.PrepareWidget.Dropdown method), 577 method), 496
- observe() (AFL.automation.prepare.SampleSeriesWidget.lomTemsg() (AFL.automation.prepare.PrepareWidget.HBox method), 585 method), 504
- observe() (AFL.automation.prepare.SampleSeriesWidget.loadsg() (AFL.automation.prepare.PrepareWidget.Label method), 592 method), 511
- observe() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 602 method), 521
- observe() (AFL.automation.prepare.SampleSeriesWidget.Text_msg() (AFL.automation.prepare.PrepareWidget.Text_method), 613 method), 534
- observe() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 620 (AFL.automation.prepare.PrepareWidget.VBox method), 542
- observe() (AFL.automation.prepare.SweepBuilderWidget.Bnttmsg() (AFL.automation.prepare.SampleSeriesWidget.Button method), 633 method), 551
- observe() (AFL.automation.prepare.SweepBuilderWidget.**6hewkly**(x) (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 641 method), 559
- observe() (AFL.automation.prepare.SweepBuilderWidget.bhBansg() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 649 method), 569
- observe() (AFL.automation.prepare.SweepBuilderWidget.bmbrdsg() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 656 method), 577
- observe() (AFL.automation.prepare.SweepBuilderWidget.baymsg() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 666 method), 585
- observe() (AFL.automation.prepare.SweepBuilderWidget. Enginesg() (AFL.automation.prepare.SampleSeriesWidget.Label method), 676 method), 593
- observe() (AFL.automation.prepare.SweepBuilderWidget.VBomsg() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 683 method), 603
- on_click() (AFL.automation.prepare.DeckBuilderWidget.Bnttmsg() (AFL.automation.prepare.SampleSeriesWidget.Text method), 387 method), 613
- on_click() (AFL.automation.prepare.PrepareWidget.Buttom_msg() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 476 method), 621
- on_click() (AFL.automation.prepare.SampleSeriesWidgetcButtnsvg() (AFL.automation.prepare.SweepBuilderWidget.Button method), 548 method), 634
- on_click() (AFL.automation.prepare.SweepBuilderWidgetCheckbox method), 631 method), 641
- on_msg() (AFL.automation.prepare.DeckBuilderWidget.Buannmsg() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 390 method), 649
- on_msg() (AFL.automation.prepare.DeckBuilderWidget.Chenklmsg() (AFL.automation.prepare.SweepBuilderWidget.Label method), 398 method), 657
- on_msg() (AFL.automation.prepare.DeckBuilderWidget.Dropdmsg() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 411 method), 667
- on_msg() (AFL.automation.prepare.DeckBuilderWidget.HBm_msg() (AFL.automation.prepare.SweepBuilderWidget.Text method), 419 method), 676
- on_msg() (AFL.automation.prepare.DeckBuilderWidget.Label_msg() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 426 method), 684

method), 154	on_trait_change() (AFL.automation.prepare.SampleSeriesWidget.Laber
on_service_state_change()	method), 593
(AFL.automation.shared.ServerDiscovery method), 78, 764, 779	.ServerDincoveryt_change() (AFL.automation.prepare.SampleSeriesWidget.Layor method), 603
	erWidgetoffextrait_change() (AFL.automation.prepare.SampleSeriesWidget.Text
method), 441	method), 613
	A dget. Tex $oldsymbol{o}$ n_trait_change() (AFL. automation. prepare. Sample Series Widget. VBox
method), 531	method), 621
	iesWidg enTexr ait_change() (AFL.automation.prepare.SweepBuilderWidget.Butto
method), 610	method), 634
**	
	derWidgenTewait_change() (AFL.automation.prepare.SweepBuilderWidget.Chec
method), 673	method), 642
on_trait_change() (AFL.automation.prepare.De method), 390	ckBuild orWidgai Buttha nge() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 649
on_trait_change()(AFL.automation.prepare.De	ckBuild orWitlgai €hehkhrge () (AFL.automation.prepare.SweepBuilderWidget.Labe
method), 398	method), 657
on_trait_change()(AFL.automation.prepare.De	ckBuild orWitigai.Drohdnige() (AFL.automation.prepare.SweepBuilderWidget.Layo
method), 411	method), 667
on_trait_change()(AFL.automation.prepare.De	ckBuilderWithgai.HBthange() (AFL.automation.prepare.SweepBuilderWidget.Text
method), 419	method), 676
	ckBuild orWithgai.Labhlange() (AFL.automation.prepare.SweepBuilderWidget.VBox
method), 426	method), 684
on_trait_change()(AFL.automation.prepare.De	
method), 436	(AFL.automation.prepare.DeckBuilderWidget.Button
on_trait_change() (AFL.automation.prepare.De	
method), 444	on_widget_constructed()
	ckBuilderWidget. (A E E C C C C C C C C C C
method), 452	static method), 399
on_trait_change() (AFL.automation.prepare.Pre	
method), 479	(AFL.automation.prepare.DeckBuilderWidget.Dropdown
on_trait_change() (AFL.automation.prepare.Pre	
method), 487	on_widget_constructed()
	epareWidget.DropdAFh.automation.prepare.DeckBuilderWidget.HBox
method), 496	static method), 419
on_trait_change() (AFL.automation.prepare.Pre	
method), 504	(AFL.automation.prepare.DeckBuilderWidget.Label
on_trait_change() (AFL.automation.prepare.Pre	
	on_widget_constructed()
- '	epareWidget.LayouAFL.automation.prepare.DeckBuilderWidget.Layout
method), 521	static method), 437
on_trait_change() (AFL.automation.prepare.Pre	
method), 534	(AFL.automation.prepare.DeckBuilderWidget.Text
$on_trait_change()$ (AFL.automation.prepare.Pre	
method), 542	on_widget_constructed()
$on_trait_change()$ (AFL.automation.prepare.San	mpleSeriesWidget. BAFibn utomation.prepare.DeckBuilderWidget.VBox
method), 551	static method), 452
<pre>on_trait_change() (AFL.automation.prepare.San</pre>	npleSeri an Widge Ch eddos tructed()
method), 559	(AFL.automation.prepare.PrepareWidget.Button
on_trait_change()(AFL.automation.prepare.San	mpleSeriesWidget. FlatatTaxt hod), 480
method), 570	on_widget_constructed()
	mpleSeriesWidget. {AF dxautomation.prepare.PrepareWidget.Checkbox
method), 577	static method), 488
on_trait_change()(AFL.automation.prepare.Sar	
method), 585	(AFL.automation.prepare.PrepareWidget.Dropdown

static method), 497	static method), 657
on_widget_constructed()	on_widget_constructed()
(AFL.automation.prepare.PrepareWidget.HBox	(AFL.automation.prepare.SweepBuilderWidget.Layout
static method), 504	static method), 667
on_widget_constructed()	on_widget_constructed()
(AFL.automation.prepare.PrepareWidget.Label	(AFL.automation.prepare.SweepBuilderWidget.Text
static method), 512	static method), 677
on_widget_constructed()	on_widget_constructed()
(AFL.automation.prepare.PrepareWidget.Layout	(AFL.automation.prepare.SweepBuilderWidget.VBox
static method), 522	static method), 684
<pre>on_widget_constructed()</pre>	OneSelectorBlowoutSampleCell (class in
(AFL.automation.prepare.PrepareWidget.Text	AFL. automation. loading. One Selector Blow out Sample Cell),
static method), 535	38, 258, 269
<pre>on_widget_constructed()</pre>	open() (AFL.automation.instrument.SeabreezeUVVis.Path
(AFL.automation.prepare.PrepareWidget.VBox	method), 200
static method), 542	open() (AFL.automation.prepare.DeckBuilderWidget.Button
<pre>on_widget_constructed()</pre>	method), 391
(AFL.automation.prepare.SampleSeriesWidget.B	uttpen() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
static method), 552	method), 399
<pre>on_widget_constructed()</pre>	open() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
(AFL. automation. prepare. Sample Series Widget. Compared to the support of the	heckbox method), 412
static method), 560	open() (AFL.automation.prepare.DeckBuilderWidget.HBox
<pre>on_widget_constructed()</pre>	method), 419
(AFL. automation. prepare. Sample Series Widget. Figure 1.000000000000000000000000000000000000	lap#n(t) (AFL.automation.prepare.DeckBuilderWidget.Label
static method), 570	method), 427
<pre>on_widget_constructed()</pre>	open() (AFL.automation.prepare.DeckBuilderWidget.Layout
(AFL. automation. prepare. Sample Series Widget. H	Box method), 437
static method), 578	open() (AFL.automation.prepare.DeckBuilderWidget.Text
on_widget_constructed()	method), 445
(AFL. automation. prepare. Sample Series Widget. In the content of the content	tapen() (AFL.automation.prepare.DeckBuilderWidget.VBox
static method), 586	method), 452
on_widget_constructed()	open() (AFL.automation.prepare.PrepareWidget.Button
(AFL. automation. prepare. Sample Series Widget. Loss of the contraction of the contrac	abel method), 480
static method), 593	open() (AFL.automation.prepare.PrepareWidget.Checkbox
on_widget_constructed()	method), 488
	appen() (AFL.automation.prepare.PrepareWidget.Dropdown
static method), 603	method), 497
on_widget_constructed()	open() (AFL.automation.prepare.PrepareWidget.HBox
(AFL. automation. prepare. Sample Series Widget. Teaching the series with the series of the series with the	
static method), 614	open() (AFL.automation.prepare.PrepareWidget.Label
on_widget_constructed()	method), 512
	Bopen() (AFL.automation.prepare.PrepareWidget.Layout
static method), 621	method), 522
on_widget_constructed()	open() (AFL.automation.prepare.PrepareWidget.Text
(AFL. automation. prepare. Sweep Builder Widget. But the substitution of the substit	
static method), 634	open() (AFL.automation.prepare.PrepareWidget.VBox
on_widget_constructed()	method), 542
	(hapkba) (AFL.automation.prepare.SampleSeriesWidget.Button
static method), 642	method), 552
on_widget_constructed()	open() (AFL.automation.prepare.SampleSeriesWidget.Checkbox
(AFL.automation.prepare.SweepBuilderWidget.H	
static method), 650	open() (AFL.automation.prepare.SampleSeriesWidget.FloatText
on_widget_constructed()	method), 570
(AFL. automation. prepare. Sweep Builder Widget. L	abpen() (AFL.automation.prepare.SampleSeriesWidget.HBox

method), 578			AFL.automation.shared.DiffractionLabeler),
open() (AFL.automation.prep	pare. Sample Series Widget. Int The Series with the series w	Text	735
method), 586		OT2Clie	nt (class in AFL.automation.prepare.OT2Client),
open() (AFL.automation.prep	pare. Sample Series Widget. Lab		62, 466, 470
method), 593			nt (class in AFL.automation.sample.CastingServer),
open() (AFL.automation.prep method), 603	oare.SampleSeriesWidget.Lay		701 t1 (AFL.automation.APIServer.APIServer.ServiceInfo
open() (AFL.automation.prep	pare.SampleSeriesWidget.Tex		attribute), 141
method), 614	•		tl (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
open() (AFL.automation.prep	pare.SampleSeriesWidget.VBo		attribute), 753
method), 621	•		tl (AFL.automation.shared.ServerDiscovery.ServiceInfo
open() (AFL.automation.prep	oare.SweepBuilderWidget.But	ton	attribute), 770
method), 634		overflo	w (AFL.automation.prepare.DeckBuilderWidget.Layout
open() (AFL.automation.prep	oare.SweepBuilderWidget.Che	eckbox	attribute), 433
method), 642		overflo	w (AFL.automation.prepare.PrepareWidget.Layout
open() (AFL.automation.prep	oare.SweepBuilderWidget.HB		attribute), 518
method), 650			พ (AFL.automation.prepare.SampleSeriesWidget.Layout
open() (AFL.automation.prep	oare.SweepBuilderWidget.Lal		attribute), 599
method), 657			w (AFL.automation.prepare.SweepBuilderWidget.Layout
open() (AFL.automation.prep	oare.SweepBuilderWidget.Lay		attribute), 663
method), 667			(AFL.automation.instrument.SeabreezeUVVis.Path
open() (AFL.automation.prep	oare.SweepBuilderWidget.Tex	t	method), 200
method), 677		D	
open() (AFL.automation.prep	oare.SweepBuilderWidget.VB	OK T	
method), 684		package	$\verb _cmd() (AFL. automation. loading. Ultimus VP ressure Controller. Ultimus VP ressure Cont$
open_instance_resource(method), 56, 363, 364
	PIServer.APIServer.Flask	padding	(AFL.automation.prepare.DeckBuilderWidget.Layout
method), 108	. A DIG A DIG		attribute), 433
open_resource()(AFL.auto	mation.APIServer.APIServer	padding	(AFL.automation.prepare.PrepareWidget.Layout
method), 121		· n	attribute), 518
openConnection() (AFL.au	tomation.toaaing.Cnemyxsyri	padding	(Aren: Widget.Layout
method), 33, 218, 22			attribute), 599
options (AFL. automation. pro	граге.Dесквинаеrwiaget.Dro	BaddIng	(AFL. automation. prepare. Sweep Builder Widget. Layout
attribute), 412	an ana Puan ana Wida at Duan da	14170	attribute), 663
attribute), 497	граге. Е гераге w шдеп. Д горао	Marent (AFL.automation.instrument.SeabreezeUVVis.Path
	ara DackRuildarWidget Lavor	ıt .	property), 202
attribute), 433	ire.DeckBuitaer wiagei.Layou	parents	(AFL.automation.instrument.SeabreezeUVVis.Path
	pare.PrepareWidget.Layout		property), 203
attribute), 518	pare.1 repare wager.Layou	parse_w	ell() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O
	are SampleSeriesWidget Lavo	Mangad	method), 61, 460, 462 addresses() (AFL.automation.APIServer.APIServer.ServiceInfo
attribute), 599	re.sampreseries wager.Eayo	rparseu_	
	are SweenRuilderWidoet Lavo) Hancad	method), 143 addresses() (AFL.automation.shared.ServerDiscovery.AsyncSer
attribute), 663	ve.sweepBunaer magen.Eayo	·parseu_	method), 755
ordinal_phase_labels()		narcod	memoa), 755 addresses() (AFL.automation.shared.ServerDiscovery.ServiceIr
	ared.DataLabelerWidget.Dat	aLabelerN	addresses() (AFL.amomanon.snarea.serverDiscovery.servicen Aodel _{ad}) 779
method), 71, 712, 72			scoped_addresses()
<pre>ordinal_phase_labels()</pre>		par seu_	(AFL.automation.APIServer.APIServer.ServiceInfo
<u>-</u>	ared.DiffractionLabeler.Diffr	actionLab	elerModel 143
method), 75, 734, 74			scoped_addresses()
OrdinalEncoder	(class in	Par sca_	(AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
AFL.automation.sha	ared.DataLabelerWidget),		method), 755
715		parsed_	scoped_addresses()
OrdinalEncoder	(class in	_	(AFL.automation.shared.ServerDiscovery.ServiceInfo

<pre>method), 772 ParseException, 383</pre>	placeholder (AFL.automation.prepare.SampleSeriesWidget.Text
_	attribute), 614
	placeholder (AFL.automation.prepare.SweepBuilderWidget.Label
AFL.automation.loading.ChemyxSyringePump),	attribute), 658
32, 217, 222	placeholder (AFL.automation.prepare.SweepBuilderWidget.Text
parser_element (AFL.automation.prepare.Component.Po	
attribute), 383	plot_binary_cb() (AFL.automation.prepare.PrepareWidget.SweepBuilde
parts (AFL.automation.instrument.SeabreezeUVVis.Path	method), 528
property), 203	plot_binary_cb() (AFL.automation.prepare.SweepBuilderWidget.Sweep
patch() (AFL.automation.APIServer.APIServer.Flask	method), 67, 670, 687
method), 121	plot_bounds() (AFL.automation.prepare.MassBalance
${\tt Path}(classinAFL.automation.instrument.SeabreezeUVV is$	
197	plot_comp() (AFL.automation.shared.DatasetWidget.DatasetWidget_View
pause() (AFL.automation.APIServer.APIServer.APIServe	
method), 17, 93, 150	plot_grid_mask() (AFL.automation.prepare.MassBalance
pause() (AFL.automation.APIServer.Client.Client	method), 372
method), 18, 153, 156	plot_sas() (AFL.automation.shared.DatasetWidget.DatasetWidget_View
pause() (AFL.automation.loading.LoadStopperDriver.Cli	
method), 235	$\verb"plot_ternary_cb") (AFL. automation. prepare. Prepare Widget. Sweep Builder and the property of the propert$
pause() (AFL.automation.prepare.DeckBuilderWidget.Cla	
method), 403	$\verb"plot_ternary_cb") (AFL. automation. prepare. Sweep Builder Widget. Sweep Builder Wid$
pause() (AFL.automation.prepare.OT2Client.Client	method), 67, 670, 687
method), 465	PneumaticPressureSampleCell (class in
pause() (AFL.automation.prepare.OT2Client.OT2Client	AFL. automation. loading. Pneumatic Pressure Sample Cell),
method), 468	39, 272, 279
pause() (AFL.automation.prepare.SampleSeriesWidget.Ca	li Pm eumaticSampleCell (class in
method), 564	AFL. automation. loading. Pneumatic Sample Cell),
<pre>pause() (AFL.automation.sample.CastingServer.Client</pre>	41, 283, 288
method), 698	pop() (AFL.automation.APIServer.Driver.PersistentConfig
${\tt pause()} \ (AFL. automation. sample. Casting Server. OT 2Clies and the control of the contr$	nt method), 161
method), 704	pop() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
$\verb"pausePump"()" (AFL. automation. loading. Chemyx Syringe Pump")" (AFL. automation. loading. Chemyx Syringe Pump") (AFL. automation. loading. Chemyx Syringe Pump") (AFL. automation. loading. Chemyx Syringe Pump") (AFL. automation. loading. loadi$	ımp.Chemy n@dund);tilōli)
method), 33, 219, 222	$\verb"pop()" (AFL. automation. loading. One Selector Blowout Sample Cell. default discovered by the property of $
permanent_session_lifetime	method), 268
(AFL.automation.APIServer.APIServer.Flask	pop() (AFL.automation.loading.PneumaticPressureSampleCell.defaultdict
attribute), 104	method), 278
PersistentConfig (class in	pop() (AFL.automation.loading.PneumaticSampleCell.defaultdict
AFL.automation.APIServer.Driver), 159	method), 288
PersistentConfig (class in	pop() (AFL.automation.loading.PushPullSelectorSampleCell.defaultdict
AFL.automation.shared.PersistentConfig),	method), 303
76, 745, 747	pop() (AFL.automation.loading.RSoXSSolutionSampleCell.defaultdict
PipetteAction (class in AFL.automation.prepare), 372	method), 314
	pop() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultdic
AFL.automation.prepare.OT2Client), 469	method), 359
	ephpk@l(AFL.automation.shared.DatasetWidget.defaultdict
attribute), 427	method), 729
	epop() (AFL.automation.shared.PersistentConfig.MutableMapping
attribute), 445	method), 744
	ulpop() (AFL.automation.shared.PersistentConfig.PersistentConfig
attribute), 512	method), 747
	expopitem() (AFL.automation.APIServer.Driver.PersistentConfig
attribute), 535	method), 162
	gptopihæm() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
attribute), 594	method), 179

- popitem() (AFL.automation.loading.OneSelectorBlowoutSpostle@xHalrfa())di&FL.automation.loading.LoadStopperDriver.LoadStoppermethod), 268 method), 240
- popitem() (AFL.automation.loading.PneumaticPressureSappqsleCekledsstruel(dicAFL.automation.loading.OneSelectorBlowoutSampleCemethod), 278 method), 258
- popitem() (AFL.automation.loading.PneumaticSampleCelpdesfuuetdetaute() (AFL.automation.loading.OneSelectorBlowoutSampleCemethod), 288 method), 262
- popitem() (AFL.automation.loading.PushPullSelectorSamphsStelbxlefcwttel(xt(AFL.automation.loading.OneSelectorBlowoutSampleCemethod), 303 method), 266
- popitem() (AFL.automation.loading.RSoXSSolutionSampleCell_defaedutie() (AFL.automation.loading.PneumaticPressureSampleCell_method), 314 method), 272
- popitem() (AFL.automation.loading.TwoSelectorBlowoutSposple@xHzdraQdiAtFL.automation.loading.PneumaticPressureSampleCelmethod), 359 method), 276
- popitem() (AFL.automation.shared.PersistentConfig.Muta**phrMapring**ute() (AFL.automation.loading.PneumaticSampleCell.PneumaticS
- popitem() (AFL.automation.shared.PersistentConfig.Persist
- port (AFL.automation.APIServer.APIServer.ServiceInfo post_execute() (AFL.automation.loading.PushPullSelectorSampleCell.Fattribute), 141 method), 300
- port (AFL.automation.shared.ServerDiscovery.AsyncServiceoute() (AFL.automation.loading.RSoXSSolutionSampleCell.Draattribute), 753 method), 307
- port (AFL.automation.shared.ServerDiscovery.ServiceInfo post_execute() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoattribute), 770 method), 311
- post() (AFL.automation.APIServer.APIServer.Flask post_execute() (AFL.automation.loading.TwoSelectorBlowoutSampleCemethod), 121 method), 352
- post_execute() (AFL.automation.APIServer.DummyDrivpoDtivexecute() (AFL.automation.prepare.Dummy_OT2_Driver.Driver method), 165 method), 457
- post_execute() (AFL:automation.APIServer.DummyDrivpoDunexpDuive() (AFL:automation.prepare.Dummy_OT2_Driver.Dummy_method), 168 method), 461
 post_execute() (AFL:automation.APIServer.DummyOT2Driver_Deverte() (AFL:automation.sample.CastingServer.CastingServer
- method), 172 method), 695
 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), 175 method), 701
- post_execute() (AFL.automation.instrument.DummySASplositerexecute() (AFL.automation.sample_env.TemperatureDeck.Driver method), 185 method), 707
- post_execute() (AFL:automation.instrument.DummySASplostningsSelSute() (AFL:automation.sample_env.TemperatureDeck.TemperatureDe
- post_execute() (AFL.automation.instrument.I22SAXS.Dnpme_execute() (AFL.automation.APIServer.Driver.Driver method), 190 method), 19, 159, 162
- post_execute() (AFL.automation.instrument.I22SAXS.I2\(\) \(
- method), 192 method), 165
 post_execute() (AFL.automation.instrument.SeabreezeUVVes_Dxbcute() (AFL.automation.APIServer.DummyDriver.DummyDriver
- method), 196

 method), 169

 post_execute() (AFL.automation.instrument.SeabreezeUVVes_Sexelowtee()) (VAFL.automation.APIServer.DummyOT2Driver.Driver_method), 206

 method), 172
- method), 206
 method), 172
 post_execute() (AFL.automation.instrument.SpecScreen_pbeivexDriver() (AFL.automation.APIServer.DummyOT2Driver.DummyDemethod), 210
 method), 175
- post_execute() (AFL.automation.instrument.SpecScreen prejectSecont for instrument.DummySAS.Driver method), 212 method), 185
- post_execute() (AFL.automation.loading.LoadStopperDprer_Driver_Dr

```
pre_execute() (AFL.automation.instrument.I22SAXS.Dripprepare_and_cast() (AFL.automation.sample.CastingServer.CastingSer
        method), 190
                                                               method), 83, 695, 705
pre_execute() (AFL.automation.instrument.I22SAXS.I22SAXS.I22SAXSTare_casting_stocks()
                                                               (AFL.automation.sample.CastingServer.CastingServer
        method), 192
pre_execute() (AFL.automation.instrument.SeabreezeUVVis.Drivermethod), 82, 695, 705
        method), 196
                                                      PrepareWidget
                                                                                    (class
                                                                                                        in
pre_execute() (AFL.automation.instrument.SeabreezeUVVis.SeabreeELUWtimation.prepare.PrepareWidget),
                                                               64, 524, 545
         method), 206
pre_execute()(AFL.automation.instrument.SpecScreen_DriepaDeWiedget_Model
                                                                                       (class
                                                                                                        in
                                                               AFL.automation.prepare.PrepareWidget),
         method), 210
pre_execute() (AFL.automation.instrument.SpecScreen_Driver.Spe6Scfe2th_Dtiver
                                                      PrepareWidget_View
         method), 212
                                                                                       (class
                                                                                                        in
pre_execute() (AFL.automation.loading.LoadStopperDriver.DriverAFL.automation.prepare.PrepareWidget),
                                                               64, 525, 545
        method), 237
pre_execute() (AFL.automation.loading.LoadStopperDripmelpRedStopperDiver
                                                                                                   module
                                                                                    (in
         method), 241
                                                               AFL.automation.prepare.PrepType), 471
pre_execute() (AFL.automation.loading.OneSelectorBlovponer$noqdes6elldQxiest()
                                                               (AFL.automation.APIServer.APIServer.Flask
        method), 258
pre_execute()(AFL.automation.loading.OneSelectorBlowoutSampleCell
                                                      PrepType (class in AFL.automation.prepare.Component),
        method), 262
pre_execute() (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwoSelectorBlowoutSampleCell
        method), 267
                                                      PrepType (class in AFL.automation.prepare.PrepType),
pre_execute() (AFL.automation.loading.PneumaticPressureSampleCell_Dr;i\d\d\d\d\)
         method), 272
                                                      PressureController
                                                                                       (class
pre_execute() (AFL.automation.loading.PneumaticPressureSample\(\mathbb{Oell.AntermationPhastdingStainpthelOallPressureController\),
        method), 276
pre_execute() (AFL.automation.loading.PneumaticSampRacesbDreController
                                                                                       (class
                                                                                                        in
                                                               AFL.automation.loading.PressureController),
        method), 282
pre_execute() (AFL.automation.loading.PneumaticSampleCell.PneumatioSampleCell
                                                      PressureController
        method), 286
                                                                                       (class
                                                                                                        in
pre_execute() (AFL.automation.loading.PushPullSelectorSampleCAlF.Daiwtomation.loading.UltimusVPressureController),
         method), 296
pre_execute() (AFL.automation.loading.PushPullSelectoPSwesplaCeADpRusbPullSAksPtwwSampleCeUtlass
                                                               AFL.automation.loading.PressureControllerAsPump),
        method), 300
pre_execute() (AFL.automation.loading.RSoXSSolutionSampleCell4Brixer, 293
        method), 307
                                                      prev_button_callback()
pre_execute() (AFL.automation.loading.RSoXSSolutionSampleCell,RFbXx88olutionSampleCell,ataLabelerWidget.DataLabelerWidget
         method), 311
                                                               method), 71, 714, 721
pre_execute() (AFL.automation.loading.TwoSelectorBlovpnerSabutta6alld2vlikback()
                                                               (AFL.automation.shared.DatasetWidget.DatasetWidget
        method), 352
pre_execute() (AFL.automation.loading.TwoSelectorBlowoutSample@thloTwoSelectorBlowoutSampleCell
         method), 357
                                                      prev_button_callback()
pre_execute() (AFL.automation.prepare.Dummy_OT2_Driver.DriveAFL.automation.shared.DiffractionLabeler.DiffractionLabeler
        method), 457
                                                               method), 75, 733, 741
pre_execute() (AFL.automation.prepare.Dummy_OT2_DpinimeRimsny(WAF_Daintamation.loading.PneumaticPressureSampleCell.P
                                                               method), 40, 275, 280
         method), 461
pre_execute() (AFL.automation.sample.CastingServer.CaptingServer() (AFL.automation.loading.PneumaticSampleCell.Pneumatic
                                                               method), 42, 285, 289
        method), 695
pre_execute() (AFL.automation.sample.CastingServer.Dpreority (AFL.automation.APIServer.APIServer.ServiceInfo
         method), 701
                                                               attribute), 141
pre_execute() (AFL.automation.sample_env.Temperatureprixtant typeAFL.automation.shared.ServerDiscovery.AsyncServiceInfo
        method), 707
                                                               attribute), 753
pre_execute() (AFL.automation.sample_env.Temperature\(\textit{DiricharFluxalDeath}\)ation.shared.ServerDiscovery.ServiceInfo
        method), 709
                                                               attribute), 770
```

```
process_components()
                                                                                                                                                                                                 query_driver() (AFL.automation.loading.LoadStopperDriver.Client
                                                                                                                                                                                                                                 method), 235
                                 (AFL.automation.prepare.MassBalance
                                method), 371
                                                                                                                                                                                                 query_driver() (AFL.automation.prepare.DeckBuilderWidget.Client
process_response() (AFL.automation.APIServer.APIServer.Flask method), 403
                                method), 125
                                                                                                                                                                                                 query_driver() (AFL.automation.prepare.OT2Client.Client
process_signal()(AFL.automation.loading.LoadStopperDriver.StopkthandlCB\65
                                                                                                                                                                                                 query_driver() (AFL.automation.prepare.OT2Client.OT2Client
                                method), 246
process_signal()(AFL.automation.loading.LoadStopperDriver.StopEthnodIC,BM28
                                method), 249
                                                                                                                                                                                                 query_driver() (AFL.automation.prepare.SampleSeriesWidget.Client
process_signal() (AFL.automation.loading.SensorCallbackThreadnSahsah)CallbackThread
                                method), 50, 330, 341
                                                                                                                                                                                                 query_driver() (AFL.automation.sample.CastingServer.Client
process_signal() (AFL.automation.loading.SensorCallbackThreadn&ihpleTbookholdCB
                                                                                                                                                                                                 query_driver() (AFL.automation.sample.CastingServer.OT2Client
                                method), 52, 333, 342
process_signal()(AFL.automation.loading.SensorCallbackThreadn&athpld)adOBv1
                                method), 51, 336, 342
                                                                                                                                                                                                 {\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), 52, 339, 342
                                                                                                                                                                                                 query_scheduler(AFL.automation.shared.ServerDiscovery.ServiceBrown
product (class in AFL.automation.prepare.factory), 691
                                                                                                                                                                                                                                 attribute), 766
                                                                                                                                                                                                 queue_state() (AFL.automation.APIServer.APIServer.APIServer
propagate_exceptions
                                                                                                                                                                                                                                 method), 16, 92, 150
                                (AFL.automation.APIServer.APIServer.Flask
                                property), 107
                                                                                                                                                                                                 queue_state() (AFL.automation.APIServer.Client.Client
properties (AFL. automation. APIS erver. APIS erver. Service Info
                                                                                                                                                                                                                                 method), 18, 153, 156
                                attribute), 143
                                                                                                                                                                                                 queue_state() (AFL.automation.loading.LoadStopperDriver.Client
properties (AFL.automation.shared.ServerDiscovery.AsyncServiceImfethod), 235
                                                                                                                                                                                                 queue_state() (AFL.automation.prepare.DeckBuilderWidget.Client
                                attribute), 755
properties (AFL.automation.shared.ServerDiscovery.ServiceInfo method), 403
                                attribute), 772
                                                                                                                                                                                                 queue_state() (AFL.automation.prepare.OT2Client.Client
PROTECTED_SAMPLE_KEYS
                                                                                                                                                                                                                                 method), 465
                                (AFL.automation.APIServer.QueueDaemon.DataTyachequstate() (AFL.automation.prepare.OT2Client.OT2Client
                                attribute), 178
                                                                                                                                                                                                                                 method), 468
PROTECTED_SYSTEM_KEYS
                                                                                                                                                                                                 queue_state() (AFL.automation.prepare.SampleSeriesWidget.Client
                                (AFL.automation.APIServer.QueueDaemon.DataTrashcan method), 564
                                attribute), 178
                                                                                                                                                                                                 queue_state() (AFL.automation.sample.CastingServer.Client
{\tt pstr}\,(AFL. automation. prepare. Component. Parse Exception
                                                                                                                                                                                                                                 method), 699
                                attribute), 383
                                                                                                                                                                                                 queue_state() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                                                      (class
PushPullSelectorSampleCell
                                                                                                                                                                                                                                 method), 704
                                                                                                                                                                                   in
                                AFL.automation.loading.PushPullSelectorSample@webled()
                                                                                                                                                                                                                                                  (AFL.automation.APIServer.Driver.Driver
                                44, 296, 303
                                                                                                                                                                                                                                 method), 19, 158, 162
                                        (AFL.automation.APIServer.APIServer.Flask queued() (AFL.automation.APIServer.DummyDriver.Driver
put()
                                method), 121
                                                                                                                                                                                                                                 method), 164
put() (AFL.automation.APIServer.APIServer.MutableQueuqueued() (AFL.automation.APIServer.DummyDriver.DummyDriver
                                method), 133
                                                                                                                                                                                                                                 method), 169
put () (AFL.automation.shared.MutableQueue.MutableQuequeued() (AFL.automation.APIServer.DummyOT2Driver.Driver
                                method), 76, 742, 743
                                                                                                                                                                                                                                 method), 171
                                                                                                                                                                                                 queued() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver
Q
                                                                                                                                                                                                                                 method), 175
\verb"qsize" () (AFL. automation. APIS erver. APIS erver. Mutable Quarter \verb"automation" (AFL. automation. instrument. Dummy SAS. Driver approximation of the property of the pro
                                                                                                                                                                                                                                 method), 184
                                 method), 133
\verb|qsize()| (AFL. automation. shared. Mutable Queue. Mutable Queue()) (AFL. automation. instrument. Dummy SAS. Dummy SAS
                                                                                                                                                                                                                                  method), 187
                                method), 76, 742, 743
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), 189
                                 method), 16, 92, 149
\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), 18, 153, 156

method), 192

```
queued() (AFL.automation.instrument.SeabreezeUVVis.Driver
                                                                                                                                                                                          22, 179, 182
                          method), 195
                                                                                                                                                               queuehandler() (AFL.automation.EpicsADLiveProcess.AreaDetectorLive
queued() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVWishod), 24, 791
                                                                                                                                                               quickbar() (AFL.automation.APIServer.Driver.Driver
                          method), 206
queued() (AFL.automation.instrument.SpecScreen_Driver.Driver
                                                                                                                                                                                         method), 19, 158, 162
                          method), 209
                                                                                                                                                               quickbar() (AFL.automation.APIServer.DummyDriver.Driver
queued() (AFL.automation.instrument.SpecScreen_Driver.SpecScreemetholder, 164
                                                                                                                                                               quickbar() (AFL.automation.APIServer.DummyDriver.DummyDriver
                          method), 212
queued() (AFL.automation.loading.LoadStopperDriver.Driver
                                                                                                                                                                                          method), 169
                                                                                                                                                               quickbar() (AFL.automation.APIServer.DummyOT2Driver.Driver
                          method), 236
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), 241
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Durithard), 175
                          method), 257
                                                                                                                                                               quickbar() (AFL.automation.instrument.DummySAS.Driver
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell. Onet/SedlectOstBlowoutSampleCell
                          method), 262
                                                                                                                                                               quickbar() (AFL.automation.instrument.DummySAS.DummySAS
queued() (AFL.automation.loading.OneSelectorBlowoutSampleCell. TwebSedlectorBlowoutSampleCell
                          method), 267
                                                                                                                                                               quickbar() (AFL.automation.instrument.I22SAXS.Driver
queued() (AFL.automation.loading.PneumaticPressureSampleCell.Dniethod), 189
                          method), 271
                                                                                                                                                               quickbar() (AFL.automation.instrument.I22SAXS.I22SAXS
queued() (AFL.automation.loading.PneumaticPressureSampleCell.Pmeuhoati)cPv2ssureSampleCell
                          method), 276
                                                                                                                                                               quickbar() (AFL.automation.instrument.SeabreezeUVVis.Driver
queued() (AFL.automation.loading.PneumaticSampleCell.Driver
                                                                                                                                                                                         method), 195
                                                                                                                                                               quickbar() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVi
                          method), 281
queued() (AFL.automation.loading.PneumaticSampleCell.PneumatiaSathple)Cell6
                          method), 286
                                                                                                                                                               quickbar() (AFL.automation.instrument.SpecScreen_Driver.Driver
queued() (AFL.automation.loading.PushPullSelectorSampleCell.Drimethod), 209
                                                                                                                                                               quickbar() (AFL.automation.instrument.SpecScreen_Driver.SpecScreen_
                          method), 295
queued() (AFL.automation.loading.PushPullSelectorSampleCell.PushPthlbSlelectdrSampleCell
                          method), 300
                                                                                                                                                               quickbar() (AFL.automation.loading.LoadStopperDriver.Driver
queued() (AFL.automation.loading.RSoXSSolutionSampleCell.Drivemethod), 236
                          method), 306
                                                                                                                                                               	extstyle{quickbar}() \ (AFL. automation. loading. Load Stopper Driver. Load Stopper Driver
queued() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXfi&hadi)nS&impleCell
                                                                                                                                                               quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Dr.
                          method), 311
queued() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.Ducithcod), 257
                                                                                                                                                               quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.On
                          method), 351
queued() (AFL.automation.loading.TwoSelectorBlowoutSampleCell. TweetSoil (2) (BlowoutSampleCell (AFL.automation.loading.TwoSelectorBlowoutSampleCell (AFL.automation.loading.TwoSelectorBlowou
                          method), 357
                                                                                                                                                               quickbar() (AFL.automation.loading.OneSelectorBlowoutSampleCell.Two
queued() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
                                                                                                                                                                                          method), 267
                          method), 456
                                                                                                                                                               quickbar() (AFL.automation.loading.PneumaticPressureSampleCell.Driv
queued() (AFL.automation.prepare.Dummy OT2 Driver.Dummy OT2thod)e271
                          method), 461
                                                                                                                                                               \verb"quickbar" () (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pneumatic Pneu
{\tt queued()}\ (AFL. automation. sample. Casting Server. Casting Server
                                                                                                                                                                                          method), 276
                                                                                                                                                               \verb"quickbar"()" (AFL. automation. loading. Pneumatic Sample Cell. Driver
                          method), 696
queued() (AFL. automation. sample. Casting Server. Driver
                                                                                                                                                                                          method), 281
                          method), 700
                                                                                                                                                               quickbar() (AFL.automation.loading.PneumaticSampleCell.PneumaticSa
queued() (AFL.automation.sample_env.TemperatureDeck.Driver
                                                                                                                                                                                          method), 286
                                                                                                                                                               quickbar() (AFL.automation.loading.PushPullSelectorSampleCell.Driver
                          method), 706
queued() (AFL.automation.sample_env.TemperatureDeck.TemperatumeDeckl), 295
                                                                                                                                                               quickbar() (AFL.automation.loading.PushPullSelectorSampleCell.PushF
                          method), 709
QueueDaemon
                                                                                     (class
                                                                                                                                                                                          method), 300
                                                                                                                                                   in
                          AFL.automation.APIServer.APIServer), 134
                                                                                                                                                               quickbar() (AFL.automation.loading.RSoXSSolutionSampleCell.Driver
QueueDaemon
                                                                                     (class
                                                                                                                                                   in
                                                                                                                                                                                          method), 306
```

quickbar() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSS

AFL.automation.APIServer.QueueDaemon),

read_load_buffer() (AFL.automation.loading.LoadStopperDriver.Sense

```
quickbar() (AFL.automation.loading.TwoSelectorBlowoutSampleCentathrively; 243
                                   method), 351
                                                                                                                                                                                                                       read_load_buffer() (AFL.automation.loading.SensorPollingThread.Sen
quickbar() (AFL.automation.loading.TwoSelectorBlowoutSampleCellEtoxBlowoftSampleCell
                                   method), 357
                                                                                                                                                                                                                        read_poll() (AFL.automation.loading.LoadStopperDriver.LoadStopperD
quickbar() (AFL.automation.prepare.Dummy OT2 Driver.Driver method), 36, 240, 251
                                                                                                                                                                                                                        read_poll_load() (AFL.automation.loading.LoadStopperDriver.LoadSto
                                   method), 456
quickbar() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_notT2od);i%or 240, 251
                                   method), 461
                                                                                                                                                                                                                        read_sensor() (AFL.automation.loading.LoadStopperDriver.LoadStoppe
quickbar() (AFL.automation.sample.CastingServer.CastingServer method), 36, 240, 250
                                   method), 696
                                                                                                                                                                                                                        read\_sensor() (AFL. automation. loading. Pneumatic Pressure Sample Cell...
quickbar() (AFL.automation.sample.CastingServer.Driver
                                                                                                                                                                                                                                                            method), 40, 276, 280
                                   method), 700
                                                                                                                                                                                                                        read_sensor() (AFL.automation.loading.PneumaticSampleCell.Pneumat
quickbar() (AFL.automation.sample_env.TemperatureDeck.Driver method), 42, 285, 289
                                                                                                                                                                                                                       read\_sensor\_poll() (AFL.automation.loading.PneumaticPressureSample)
                                   method), 706
quickbar() (AFL.automation.sample_env.TemperatureDeck.TemperatureDeck.726, 280
                                                                                                                                                                                                                       read_sensor_poll() (AFL.automation.loading.PneumaticSampleCell.Pn
                                   method), 709
quickbar_test() (AFL.automation.APIServer.DummyDriver.DummyDthveh), 42, 286, 289
                                   method), 21, 168, 170
                                                                                                                                                                                                                       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), 22, 174, 176
                                                                                                                                                                                                                                                            method), 40, 277, 280
quickbar_test2()(AFL.automation.APIServer.DummyDnieardDnensonDrivelt_load()
                                   method), 21, 168, 170
                                                                                                                                                                                                                                                            (AFL.automation.loading.PneumaticSampleCell.PneumaticSamp
quickbar_test2() (AFL.automation.APIServer.DummyOT2Driver.DummyDr4Ver286, 289
                                   method), 22, 174, 176
                                                                                                                                                                                                                       read_temp() (AFL.automation.sample_env.TemperatureDeck.Temperature
                                                                                                                                                                                                                                                            method), 69, 709, 710
R
                                                                                                                                                                                                                        read_text() (AFL.automation.instrument.SeabreezeUVVis.Path
ramp_dispense() (AFL.automation.loading.DigitalOutPressureConffellerDigitalOutPressureController
                                                                                                                                                                                                                        readlink() (AFL.automation.instrument.SeabreezeUVVis.Path
                                   method), 224
ramp_dispense() (AFL.automation.loading.DigitalOutPressureConffeller!PressureController
                                                                                                                                                                                                                        redirect() (AFL.automation.APIServer.APIServer.Flask
                                   method), 225
ramp_dispense() (AFL.automation.loading.PressureController.PressureController.
                                                                                                                                                                                                                        reduced() (AFL. automation. instrument. Seabreeze UVV is. Seabreeze UVV is
                                    method), 42, 290, 291
ramp_dispense() (AFL.automation.loading.UltimusVPressureController
                                                                                                                                                                                                                       ReduceDaemon
                                                                                                                                                                                                                                                                                                                                              (class
                                   method), 362
{\tt ramp\_dispense()} \ (AFL. automation. loading. Ultimus VP ressure Controller. {\tt QUARMASIOP} representation of the controller. {\tt Q
                                                                                                                                                                                                                                                             24, 793, 796
                                    method), 363
read()(AFL.automation.loading.LoadStopperDriver.SensoFP9HstePurblueprint()
                                                                                                                                                                                                                                                            (AFL.automation.APIServer.APIServer.Flask
                                   method), 243
                                                                                                                                                                                                                                                            method), 111
read() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                       register_error_handler()
                                   method), 49, 322, 327
                                                                                                                                                                                                                                                             (AFL.automation.APIServer.APIServer.Flask
read() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                                                            method), 121
                                   method), 49, 325, 328
                                                                                                                                                                                                                       register\_service() (AFL.automation.APIServer.APIServer.Zeroconf
read()
                                                              (AFL.automation.loading.Sensor.Sensor
                                                                                                                                                                                                                                                            method), 147
                                   method), 49, 327
\verb"read()" (AFL. automation. loading. Sensor Polling Thread. Sensor
                                                                                                                                                                                                                                                             method), 776
                                   method), 53, 344, 346
                                                           (AFL. automation. prepare. Component DB \quad \texttt{relative\_to()} \ (AFL. automation. instrument. Seabreeze UVV is. Pathology of the property of th
read()
                                                                                                                                                                                                                                                             method), 203
                                   method), 368
\verb|read_bytes()| (AFL. automation. instrument. Seabreeze UVV \texttt{E-Parse}()| (AFL. automation. APIServer. APIServer. File Handler than the seabreeze UVV \texttt{E-Parse}()| (AFL. automation. APIServer. APIServer. File Handler than the seabreeze UVV \texttt{E-Parse}()| (AFL. automation. APIServer. APIServer. APIServer. File Handler than the seabreeze UVV \texttt{E-Parse}()| (AFL. automation. APIServer. APISER. APISER.
                                                                                                                                                                                                                                                             method), 97
                                    method), 200
{\tt read\_integrated()}\ (AFL. automation. instrument. I22SAX \$. \ref{2.234} \$. (AFL. automation. APIS erver. APIS erver. SMTPH and lertical and the substitution of th
                                                                                                                                                                                                                                                             method), 138
                                   method), 26, 192, 193
                                                                                                                                                                                                                        reload() (AFL.automation.shared.widgetui.Markdown
```

method), 311

method), 788	method), 614
${\tt remove()}$ (AFL.automation.APIServer.APIServer.Mutable ${\tt Qn}$	emove_class() (AFL.automation.prepare.SampleSeriesWidget.VBox
method), 133	method), 622
remove() (AFL.automation.prepare.ComponentDB r	emove_class() (AFL.automation.prepare.SweepBuilderWidget.Button
method), 368	method), 634
${\tt remove()}$ (AFL.automation.shared.Mutable ${\it Q}$ ueue.Mutable ${\it Q}$	hmove_class() (AFL.automation.prepare.SweepBuilderWidget.Checkb
method), 76, 742, 743	method), 642
	emove_class() (AFL.automation.prepare.SweepBuilderWidget.HBox
(AFL.automation.APIServer.APIServer.Zeroconf	method), 650
· · · · · · · · · · · · · · · · · · ·	emove_class() (AFL.automation.prepare.SweepBuilderWidget.Label
remove_all_service_listeners()	method), 658
(AFL.automation.shared.ServerDiscovery.Zerocom	emove_class() (AFL.automation.prepare.SweepBuilderWidget.Text
method), 776	method), 677
remove_class()(AFL.automation.prepare.DeckBuilderWin	(gmbBattohass() (AFL.automation.prepare.SweepBuilderWidget.VBox
method), 391	method), 685
remove_class()(AFL.automation.prepare.DeckBuilderWin	(mn)Cheiklam() (AFL.automation.APIServer.APIServer.APIServer
method), 399	method), 17, 93, 150
remove_class()(AFL.automation.prepare.DeckBuilderWin	(emoDeopdemh) (AFL.automation.APIServer.Client.Client
method), 412	method), 18, 153, 156
remove_class()(AFL.automation.prepare.DeckBuilderWin	gmoHBoixtem() (AFL.automation.loading.LoadStopperDriver.Client
method), 419	method), 235
remove_class()(AFL.automation.prepare.DeckBuilderWin	emokebaltem() (AFL.automation.prepare.DeckBuilderWidget.Client
method), 427	method), 403
remove_class()(AFL.automation.prepare.DeckBuilderWin	emoVextitem() (AFL.automation.prepare.OT2Client.Client
method), 445	method), 465
remove_class()(AFL.automation.prepare.DeckBuilderWin	emoVBoitem() (AFL.automation.prepare.OT2Client.OT2Client
method), 453	method), 469
remove_class()(AFL.automation.prepare.PrepareWidget x	Pantowe_item() (AFL.automation.prepare.SampleSeriesWidget.Client
method), 480	method), 564
remove_class()(AFL.automation.prepare.PrepareWidgetm	Elmoxkbo item() (AFL.automation.sample.CastingServer.Client
method), 488	method), 699
remove_class()(AFL.automation.prepare.PrepareWidget x	@mqxt@virt.em() (AFL.automation.sample.CastingServer.OT2Client
method), 497	method), 704
remove_class()(AFL.automation.prepare.PrepareWidget x	Hove_items()(AFL.automation.APIServer.APIServer.APIServer
method), 504	method), 17, 93, 150
remove_class()(AFL.automation.prepare.PrepareWidget x	(antown e_listener() (AFL.automation.APIServer.APIServer.Zeroconf
method), 512	method), 148
remove_class()(AFL.automation.prepare.PrepareWidgetfi	${f egin{array}{c} {f Emo}ve_listener() \ (AFL. automation. shared. Server Discovery. Zero confusion of the confusion of $
method), 535	method), 778
remove_class()(AFL.automation.prepare.PrepareWidget r	$oldsymbol{e}$ flow ve_service() (AFL. automation. shared. Server Discovery. As ync Zero
method), 543	method), 759
remove_class()(AFL.automation.prepare.SampleSeriesWa	dycovieuservice_listener()
method), 552	(AFL.automation.APIServer.APIServer.Zeroconf
${\tt remove_class()}\ (AFL. automation. prepare. Sample Series William Series Wil$	dget.Che nkhho d), 147
method), 560	emove_service_listener()
remove_class()(AFL.automation.prepare.SampleSeriesWi	dget.Flo@AFEkt.automation.shared.ServerDiscovery.Zeroconf
method), 570	method), 776
remove_class()(AFL.automation.prepare.SampleSeriesWa	dpervldBotock_cb() (AFL.automation.prepare.PrepareWidget.StockBuil
method), 578	method), 528
remove_class()(AFL.automation.prepare.SampleSeriesWa	dwerden_Evock_cb() (AFL.automation.prepare.StockBuilderWidget.Stock
method), 586	method), 66, 626, 628
remove_class()(AFL.automation.prepare.SampleSeriesWa	dywardealbudrtical_lines()
method), 594	(AFL. automation. shared. Data Labeler Widget. Data Labeler View

 ${\tt remove_class()}\ (AFL. automation. prepare. Sample Series Widget. Textmethod), 71, 712, 722$

```
remove_vertical_lines()
                                                                                                                                                                                                                                                                      method), 36, 240, 251
                                      (AFL.automation.shared.DiffractionLabeler.DiffraceiserLabreterViewgets()
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Dr
                                    method), 75, 735, 741
Removed (AFL.automation.shared.ServerDiscovery.ServiceStateChangeethod), 61, 459, 461
                                      attribute), 773
                                                                                                                                                                                                                                reset_pump() (AFL.automation.loading.PneumaticSampleCell.Pneumatic
removeFilter() (AFL.automation.APIServer.APIServer.FileHandlemethod), 41, 285, 289
                                                                                                                                                                                                                                reset_queue_daemon()
                                    method), 97
removeFilter() (AFL.automation.APIServer.APIServer.SMTPHandlAFL.automation.APIServer.APIServer.APIServer
                                     method), 138
                                                                                                                                                                                                                                                                      method), 16, 91, 149
rename()(AFL.automation.instrument.SeabreezeUVVis.Patteset_queue_daemon()
                                                                                                                                                                                                                                                                      (AFL.automation.APIServer.Client.Client
                                    method), 201
rename_component()(AFL.automation.prepare.factory.Solution
                                                                                                                                                                                                                                                                     method), 18, 153, 156
                                    method), 690
                                                                                                                                                                                                                                 reset_queue_daemon()
rename_component()(AFL.automation.prepare.Solution
                                                                                                                                                                                                                                                                      (AFL.automation.loading.LoadStopperDriver.Client
                                     method), 377
                                                                                                                                                                                                                                                                       method), 235
render_template()
                                                                                                                                  (in
                                                                                                                                                                                           module reset_queue_daemon()
                                    AFL.automation.APIServer.APIServer), 86
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.DeckBuilderWidget.Client
render_unqueued() (AFL.automation.APIServer.APIServer.APIServaethod), 403
                                    method), 16, 92, 150
                                                                                                                                                                                                                                reset_queue_daemon()
reorder_queue() (AFL.automation.APIServer.APIServer.APIServer(AFL.automation.prepare.OT2Client.Client
                                    method), 17, 93, 150
                                                                                                                                                                                                                                                                      method), 465
replace() (AFL.automation.instrument.SeabreezeUVVis.Pndset_queue_daemon()
                                                                                                                                                                                                                                                                      (AFL.automation.prepare.OT2Client.OT2Client
                                     method), 201
request() (AFL.automation.APIServer.APIServer.ServiceInfo
                                                                                                                                                                                                                                                                      method), 469
                                                                                                                                                                                                                                 reset_queue_daemon()
                                    method), 143
request() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfAFL.automation.prepare.SampleSeriesWidget.Client
                                     method), 755
                                                                                                                                                                                                                                                                      method), 564
request() (AFL.automation.shared.ServerDiscovery.Servirels@t_queue_daemon()
                                    method), 772
                                                                                                                                                                                                                                                                      (AFL.automation.sample.CastingServer.Client
request_class(AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                      method), 698
                                     attribute), 103
                                                                                                                                                                                                                                 reset_queue_daemon()
{\tt request\_context()} \ (AFL. automation. APIS erver. APIS erver. Flask \ \ (AFL. automation. sample. Casting Server. OT 2 Client 1 and 
                                     method), 126
                                                                                                                                                                                                                                                                      method), 704
reset() (AFL.automation.APIServer.QueueDaemon.DataTresetusample() (AFL.automation.APIServer.Driver.Driver
                                     method), 179
                                                                                                                                                                                                                                                                      method), 19, 159, 162
reset() (AFL. automation. Epics ADLive Process. Client. Clienteset_sample() (AFL. automation. APIS erver. Dummy Driver. Dri
                                    method), 24, 792, 793
                                                                                                                                                                                                                                                                      method), 165
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), 36, 240, 251
                                                                                                                                                                                                                                                                      method), 169
reset()
                                                                   (AFL.automation.prepare.MassBalance reset_sample() (AFL.automation.APIServer.DummyOT2Driver.Driver
                                                                                                                                                                                                                                                                     method), 172
                                    method), 371
                                                                  (AFL. automation. prepare. Sample Series reset\_sample() (AFL. automation. API Server. Dummy OT2 Driver. Dummy III) and the sample of the sam
reset()
                                                                                                                                                                                                                                                                      method), 175
                                    method), 374
reset_dataset() (AFL.automation.shared.DatasetWidgetdestetsetStindgete() (AFL.automation.APIServer.QueueDaemon.DataTrashca
                                     method), 73, 725, 731
                                                                                                                                                                                                                                                                      method), 179
reset_dataset() (AFL.automation.shared.DatasetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDastersetWidgetsDasterset
                                     method), 73, 726, 731
                                                                                                                                                                                                                                                                      method), 185
reset_load_buffer()
                                                                                                                                                                                                                                reset_sample() (AFL.automation.instrument.DummySAS.DummySAS
                                      (AFL.automation.loading.LoadStopperDriver.SensorPollingThethad), 187
                                                                                                                                                                                                                                \verb"reset_sample()" (AFL. automation. instrument. I22SAXS. Driver)" and the sample () is a simple of the sample of
                                    method), 243
reset_load_buffer()
                                                                                                                                                                                                                                                                      method), 190
                                    (AFL.automation.loading.SensorPollingThread.SensorPollingThread.SensorPollingThread.AFL.automation.instrument.I22SAXS.I22SAXS
                                    method), 53, 344, 346
                                                                                                                                                                                                                                                                     method), 192
reset_pol1() (AFL.automation.loading.LoadStopperDrivarelsardStampdel())i(AFL.automation.instrument.SeabreezeUVVis.Driver
```

method), 371

J. D. 106	d D 26 241 251
method), 196 reset_sample() (AFL.automation.instrument.SeabreezeUYVieSede	method), 36, 241, 251
method), 206	(AFL.automation.loading.OneSelectorBlowoutSampleCell.OneSe
reset_sample() (AFL.automation.instrument.SpecScreen_Driver.I	,
	tank_levels()
reset_sample() (AFL.automation.instrument.SpecScreen_Driver.S	
method), 212	method), 265
reset_sample() (AFL.automation.loading.LoadStopperDnieselDri	
method), 237	(AFL.automation.loading.PneumaticPressureSampleCell.Pneuma
reset_sample() (AFL.automation.loading.LoadStopperDriver.Load	
	tank_levels()
reset_sample() (AFL.automation.loading.OneSelectorBlowoutSat	
method), 258	method), 41, 285, 289
reset_sample() (AFL.automation.loading.OneSelectorBlovesuerSa	
method), 262	AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolut
reset_sample() (AFL.automation.loading.OneSelectorBlowoutSan	
	tank_levels()
method), 267 reset_sample() (AFL.automation.loading.PneumaticPressureSam	· · · · · · · · · · · · · · · · · · ·
	• •
method), 272	method), 55, 356, 361
reset_sample() (AFL.automation.loading.PneumaticPressesSum	
method), 276	method), 370
reset_sample() (AFL.automation.loading.PneumaticSampleSextLi	
method), 282	method), 371
reset_sample() (AFL.automation.loading.PneumaticSampleSextLi	
method), 286	method), 61, 460, 462
reset_sample() (AFL.automation.loading.PushPullSelectveScenup	
method), 296	method), 62, 468, 470
reset_sample() (AFL.automation.loading.PushPullSelectveScenup	
method), 300	method), 703
reset_sample() (AFL.automation.loading.RSoXSSolution Sampl e(
method), 307	method), 527
reset_sample() (AFL.automation.loading.RSoXSSolution Sample(
method), 311	method), 65, 607, 624
reset_sample() (AFL.automation.loading.TwoSelectorBlamesaSun	
method), 352	method), 200
reset_sample() (AFL.automation.loading.TwoSelectorBlanesmonn	
method), 357	attribute), 103
reset_sample()(AFL.automation.prepare.Dummy_OT2_DesearD	
method), 457	method), 33, 219, 222
$\verb reset_sample() (AFL. automation. prepare. Dummy_OT2_\textit{Deturio}) $	hanohyj QTQA_FIriovatomation.APIServer.APIServer.APIServer
method), 461	method), 17, 92, 150
${\tt reset_sample()}\ (AFL. automation. sample. Casting Server. {\tt Constitutive})$	vevobj() (AFL.automation.APIServer.Client.Client
method), 696	method), 18, 153, 156
${\tt reset_sample()}\ (AFL. automation. sample. Casting Server. {\tt Dreiteo} ieven the content of the content of$	ve_obj() (AFL.automation.APIServer.Driver.Driver
method), 701	method), 20, 159, 163
${\tt reset_sample()}\ (AFL. automation. sample_env. Temperature {\tt Decket})$	Deivobj() (AFL.automation.APIServer.DummyDriver.Driver
method), 707	method), 165
reset_sample() (AFL.automation.sample_env.TemperatureDerkel	ն <mark>արջեց (փ</mark> re A)E Ekautomation.APIServer.DummyDriver.DummyDrive
method), 709	method), 169
	ve_obj()(AFL.automation.APIServer.DummyOT2Driver.Driver
method), 370	method), 172
	ve_obj()(AFL.automation.APIServer.DummyOT2Driver.DummyI

874 Index

reset_stopper() (AFL.automation.loading.LoadStopperDreiterilerredStopperDreiterredStoppe

method), 175

method), 185	method), 465
retrieve_obj() (AFL.automation.instrument.Dummy.	SAS Petminy &A_S obj() (AFL.automation.prepare.OT2Client.OT2Client
method), 187	method), 469
retrieve_obj() (AFL.automation.instrument.I22SAX	S.Drivetrieve_obj() (AFL.automation.prepare.SampleSeriesWidget.Client
method), 190	method), 564
retrieve_obj() (AFL.automation.instrument.I22SAX	S.12 254XS eve_obj() (AFL.automation.sample.CastingServer.CastingServer
method), 193	method), 696
	zeU Y&xDene robj() (AFL.automation.sample.CastingServer.Client
method), 196	method), 699
retrieve_obj() (AFL.automation.instrument.Seabree	zeU Y&xSexlereobejUYVAF L.automation.sample.CastingServer.Driver
method), 206	method), 701
	een <u>Adtivir Dejo</u> rbj() (AFL.automation.sample.CastingServer.OT2Client
method), 210	method), 704
retrieve_obj() (AFL.automation.instrument.SpecScr	een_ 10tivir.Spe_03ajten_AVilivan tomation.sample_env.TemperatureDeck.Driver
method), 212	method), 707
retrieve_obj() (AFL.automation.loading.LoadStoppe	erD niearChivn t_obj() (AFL.automation.sample_env.TemperatureDeck.Tempera
method), 235	method), 709
retrieve_obj() (AFL.automation.loading.LoadStoppe	erDniveneDti(ver(AFL.automation.APIServer.Driver.PersistentConfig
method), 237	method), 161
retrieve_obj() (AFL.automation.loading.LoadStoppe	erDnienekradStApplenDnineation.shared.PersistentConfig.PersistentConfig
method), 241	method), 77, 747, 748
$\verb"retrieve_obj()" (AFL. automation. loading. One Selector of the property of$	orBlo newalSedytloKetLDriad er()
method), 258	(AFL.automation.APIServer.APIServer.JWTManager
$\verb"retrieve_obj()" (AFL. automation. loading. One Selector of the property of$	orBlowoutSam pleGeld.)O nSelectorBlowoutSampleCell
method), 262	RFC
$\verb"retrieve_obj()" (AFL. automation. loading. One Selector of the property of$	าrBlowou เรเต ก ุของ ะปุดรัพoSelectorBlowoutSampleCell
method), 267	${\tt rglob()}\ (AFL. automation. instrument. Seabreeze UVV is. Path$
retrieve_obj() (AFL.automation.loading.Pneumatic	PressureSampl n&hld) ri3@0
method), 272	${\tt right} (AFL. automation. prepare. Deck Builder Widget. Layout$
retrieve_obj() (AFL.automation.loading.Pneumatic	PressureSamp laGebluPe) อุณิฑิสticPressureSampleCell
method), 276	${\tt right} (AFL. automation. prepare. Prepare Widget. Layout$
retrieve_obj() (AFL.automation.loading.Pneumatics	SampleCell.Dristeribute), 518
method), 282	${\tt right} (AFL. automation. prepare. Sample Series Widget. Layout$
retrieve_obj() (AFL.automation.loading.Pneumatics	SampleCell.PneturilatieScampleCell
method), 286	${\tt right} (AFL. automation. prepare. Sweep Builder Widget. Layout$
$\verb"retrieve_obj()" (AFL. automation. loading. PushPullSet and the property of the property of$	electorSample Gulli Dinie)e,1663
method), 296	${\tt rinseAll()} \ (AFL. automation. loading. One Selector Blow out Sample Cell. On the contraction of the c$
$\verb"retrieve_obj" () \textit{ (AFL. automation. loading. PushPullSet} \\$	electorSample ઈપ્સીપ્રેમિઝેત્રે\પ્રેમ્પિડિ electorSampleCell
method), 300	${\tt rinseAll()} \ (AFL. automation. loading. One Selector Blow out Sample Cell. Two the control of the control$
$\verb"retrieve_obj()" (AFL. automation. loading. RSoXSS olumbration of the property of the prope$	tionSampleCebhdDrively; 266
method), 307	${\tt rinseAll()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. Pneumatic $
$\verb"retrieve_obj()" (AFL. automation. loading. RSoXSS olumbrates a loading. ASoXSS olumbrates a loading$	tionSampleCe hlสเรือส ังรูSเป็นนี้อีกรูSนิเคยโยCell
method), 311	${\tt rinseAll()} \ (AFL. automation. loading. Pneumatic Sample Cell. Pneumatic Sample Cell.$
$\verb"retrieve_obj()" (AFL. automation. loading. Two Selectors and the property of the property $	rBlowoutSam pheCheld DAive285, 289
method), 352	${\tt rinseAll()} \ (AFL. automation. loading. Push Pull Selector Sample Cell. Push Pull Select$
${\tt retrieve_obj()}\ (AFL. automation. loading. Two Selecto$	rBlowoutSam pleCxdlXx45 \$&l@&t&@WlowoutSampleCell
method), 357	${\tt rinseAll()} \ (AFL. automation. loading. RSoXSS olution Sample Cell. RSoXSS of the control $
<pre>retrieve_obj() (AFL.automation.prepare.DeckBuilde</pre>	erWidget.Cliennethod), 47, 311, 316
method), 403	$\verb rinseAll() (AFL. automation. loading. Two Selector Blow out Sample Cell. Two Selectors and Sel$
<pre>retrieve_obj() (AFL.automation.prepare.Dummy_O</pre>	
method), 457	rinseCatch() (AFL.automation.loading.OneSelectorBlowoutSampleCell.
retrieve_obj()(AFL.automation.prepare.Dummy_O	
method), 461	rinseCatch() (AFL.automation.loading.OneSelectorBlowoutSampleCell.

method), 266

 $\verb"retrieve_obj()" (AFL. automation. prepare. OT 2C lient. Client$

```
rinseCatch() (AFL.automation.loading.PushPullSelectors@oople@ath.PASHP.automSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelectionSathfSelec
                            method), 45, 301, 305
                                                                                                                                                                                                      attribute), 123
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), 46, 311, 316
                                                                                                                                                                                                      method), 121
rinseCatch() (AFL.automation.loading.TwoSelectorBlow&SS&SpddCttlich&Ss&pdaCeBlowoutSampleCssll
                           method), 56, 357, 361
                                                                                                                                                                                                     AFL.automation.loading.RSoXSSolutionSampleCell),
rinseCell() (AFL.automation.loading.OneSelectorBlowoutSampleCell, @neSelectorBlowoutSampleCell
                                                                                                                                                                         run() (AFL.automation.APIServer.APIServer.APIServer
                           method), 262
rinseCell() (AFL.automation.loading.OneSelectorBlowoutSampleGnldtflod)Sellect02BldwoutSampleCell
                                                                                                                                                                                                            (AFL.automation.APIServer.APIServer.Flask
                           method), 266
                                                                                                                                                                         run()
rinseCell() (AFL.automation.loading.PneumaticPressureSampleColleHundi)mattePressureSampleCell
                            method), 40, 275, 280
                                                                                                                                                                         run() (AFL.automation.APIServer.APIServer.QueueDaemon
rinseCell() (AFL.automation.loading.PneumaticSampleCell.PneumaathSaimpleCell
                           method), 41, 285, 289
                                                                                                                                                                         run() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
rinseCell() (AFL.automation.loading.PushPullSelectorSampleCelltRetslot\(\text{in}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{l}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text{c}\)\(\text
                            method), 45, 299, 305
                                                                                                                                                                         run() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaem
rinseCell() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXISSolutionSampleCell
                           method), 46, 310, 316
                                                                                                                                                                         run() (AFL.automation.loading.LoadStopperDriver.SensorPollingThread
rinseCell() (AFL.automation.loading.TwoSelectorBlowoutSampleGulttlindffelectorBlowoutSampleCell
                           method), 56, 357, 361
                                                                                                                                                                         run() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv1
rinseCellFlood() (AFL.automation.loading.OneSelectorBlowoutSampheQellQmeSelectorBlowoutSampleCell
                           method), 262
                                                                                                                                                                         run() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
rinseCellFlood() (AFL.automation.loading.OneSelectorBlowoutSampheQellZMoSelectorBlowoutSampheCell
                                                                                                                                                                         run() (AFL.automation.loading.Sensor.DummySensor1
                            method), 266
rinseCellFlood() (AFL.automation.loading.PushPullSelectorSampleCell PushPullSelectorSampleCell
                           method), 45, 300, 305
                                                                                                                                                                         run() (AFL.automation.loading.Sensor.DummySensor2
rinseCellFlood() (AFL.automation.loading.RSoXSSolutionSampleGelh&SnXSSolutionSampleCell
                           method), 46, 311, 316
                                                                                                                                                                         {\tt run()}\ (AFL. automation. loading. Sensor Callback Thread. Sensor Callbac
rinseCellFlood() (AFL.automation.loading.TwoSelectorBlowoutSampleQell5TwoSelectorBlowoutSampleCell
                           method), 56, 357, 361
                                                                                                                                                                         run() (AFL.automation.loading.SensorCallbackThread.SimpleThresholdC.
rinseCellPull() (AFL.automation.loading.OneSelectorBlowoutSampltfOell),OneSelectorBlowoutSampleCell
                            method), 262
                                                                                                                                                                         run() (AFL. automation. loading. Sensor Callback Thread. Stop Load CBv1
rinseCellPull() (AFL.automation.loading.OneSelectorBlowoutSampltf@ell], TradSelectorBlowoutSampleCell
                            method), 266
                                                                                                                                                                         run() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
rinseCellPull() (AFL.automation.loading.PushPullSelectorSampleCellUshPullSelectorSampleCell
                           method), 45, 300, 305
                                                                                                                                                                         run() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
rinseCellPull() (AFL.automation.loading.RSoXSSolutionSampleGelltRSddX$SolutionStimpleCell
                            method), 46, 310, 316
                                                                                                                                                                         run() (AFL.automation.shared.DataLabelerWidget.DataLabelerView
rinseCellPull() (AFL.automation.loading.TwoSelectorBlowoutSampltfGell,TMqSellectOrBlowoutSampleCell
                           method), 56, 357, 361
                                                                                                                                                                         run() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
rinseSyringe() (AFL.automation.loading.OneSelectorBlowoutSampleCiell.QneSelectorBlowoutSampleCiell
                            method), 262
                                                                                                                                                                         run() (AFL.automation.shared.DatasetWidget.DatasetWidget
rinseSyringe() (AFL.automation.loading.OneSelectorBlowoutSampleCiell\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time\)\(\time
                                                                                                                                                                         run() (AFL.automation.shared.DatasetWidget.DatasetWidget_View
                           method), 266
rinseSyringe() (AFL.automation.loading.PushPullSelectorSample@edlhDd\hP4\llSe\&cto\SampleCell
                            method), 45, 299, 305
                                                                                                                                                                         run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabeler
rinseSyringe() (AFL.automation.loading.RSoXSSolutionSampleCethARSoXSSolutionAStampleCell
                           method), 46, 310, 316
                                                                                                                                                                         run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView
rinseSyringe() (AFL.automation.loading.TwoSelectorBlowoutSampleCell J.WoSelectorBlowoutSampleCell
                            method), 56, 356, 361
                                                                                                                                                                         run() (AFL.automation.shared.ServerDiscovery.RunThread
rmdir() (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                      method), 78, 761, 779
                                                                                                                                                                         run() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                           method), 201
{\tt root} (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                      method), 768
                                                                                                                                                                         run_threaded() (AFL.automation.APIServer.APIServer.APIServer
                           property), 203
```

method), 16, 92, 149 RunThread (class in AFL.automation.shared.ServerDiscove	save_cb() (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWidgry), method), 60, 404, 455
78, 760, 779	save_cb() (AFL.automation.prepare.PrepareWidget.DeckBuilderWidget method), 490
S	save_cb() (AFL.automation.prepare.PrepareWidget.StockBuilderWidget
sa_aio_discover_server_by_name()	method), 528
(AFL.automation.APIServer.Client.ServerDiscove class method), 154	save_cb() (AFL.automation.prepare.StockBuilderWidget.StockBuilderwidget.StockBuilderwi
sa_aio_discover_server_by_name()	SeabreezeUVVis (class in
(AFL automation shared Server Discovery Server)	Discovery AFL.automation.instrument.SeabreezeUVVis),
class method), 79, 764, 779	27, 203, 206
sa_discover_server_by_name()	secret_key (AFL.automation.APIServer.APIServer.Flask
(AFL.automation.APIServer.Client.ServerDiscove	ry attribute), 104
class method), 155	select_jinja_autoescape()
sa_discover_server_by_name()	(AFL.automation.APIServer.APIServer.Flask
(AFL.automation.shared.ServerDiscovery.ServerI	Discovery method), 109
class method), 79, 764, 779	$\verb selectPort() (AFL. automation. loading. Cetoni Multi Pos Valve. Cetoni M$
SainSmartRelay (class in	method), 31, 215, 216
AFL.automation.loading.SainSmartRelay), 47, 318, 319	selectPort() (AFL.automation.loading.CetoniMultiPosValve.FlowSelector method), 216
samefile()(AFL.automation.instrument.SeabreezeUVVis.	$\textbf{palle} \textbf{ctPort()} \ (AFL. automation. loading. Double Vici Multipos Selector. Double Vic$
method), 200	method), 34, 226, 229
Sample (class in AFL.automation.prepare), 373	$\verb selectPort() (AFL. automation. loading. Double Vici Multipos Selector. Flow and the property of the prope$
<pre>sample_composition_space()</pre>	method), 227
$(AFL. automation. prepare. Mass Balance \ % AFL. automation. Prepa$	selectPort() (AFL. automation. loading. Double Vici Multipos Selector. Vicing Selector.
method), 372	method), 228
277	हर्ड किन्स्य क्रियार्ग्स्ट्रि.automation.loading.FlowSelector.FlowSelector method), 35, 232
286	an heat April () (AFL automation loading. ViciMultipos Selector. Flow Selector method), 364
301	salest Repret (MAFL.automation.loading.ViciMultiposSelector.ViciMultipo method), 57, 366
SampleCell (class in AFL.automation.loading.RSoXSSolut. 312	SANGAD JECTELY, automation. APIServer. APIServer. Zeroconf method), 148
SampleCell (class in AFL.automation.loading.SampleCell) 48, 320, 321	send() (AFL.automation.prepare.DeckBuilderWidget.Button method), 391
SampleCell (class in AFL automation loading TwoSelector	Bandsin Samplactemation.prepare.DeckBuilderWidget.Checkbox
352	method), 399
SampleSeries (class in AFL.automation.prepare), 373	send() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
SampleSeriesTool_reset_cb()	method), 412
(AFL.automation.prepare.PrepareWidget.Prepare method), 64, 524, 545	mend(y) (AFL.automation.prepare.DeckBuilderWidget.HBox method), 419
SampleSeriesWidget (class in	send() (AFL.automation.prepare.DeckBuilderWidget.Label
AFL.automation.prepare.PrepareWidget),	method), 427
525	send() (AFL.automation.prepare.DeckBuilderWidget.Layout
SampleSeriesWidget (class in	method), 437
AFL.automation.prepare.SampleSeriesWidget), 65, 605, 624	send() (AFL.automation.prepare.DeckBuilderWidget.Text method), 445
SampleSeriesWidget_Model (class in	send() (AFL.automation.prepare.DeckBuilderWidget.VBox
AFL.automation.prepare.SampleSeriesWidget),	method), 453
66, 607, 624	send() (AFL.automation.prepare.PrepareWidget.Button
SampleSeriesWidget_View (class in	method), 480
AFL.automation.prepare.SampleSeriesWidget), 66, 607, 624	send() (AFL.automation.prepare.PrepareWidget.Checkbox method), 488

send() (AFL.automation.prepare.PrepareWidget.Dropdown method), 60, 404, 455
method), 497 send_deck_cb() (AFL.automation.prepare.PrepareWidget.DeckBuilder
send() (AFL.automation.prepare.PrepareWidget.HBox method), 490
method), 504 send_deck_config() (AFL.automation.prepare.Deck
send() (AFL.automation.prepare.PrepareWidget.Label method), 370
method), 512 send_deck_config() (AFL.automation.prepare.DeckBuilderWidget.De
send() (AFL.automation.prepare.PrepareWidget.Layout method), 60, 405, 455
method), 522 send_file() (in module
send() (AFL.automation.prepare.PrepareWidget.Text
······································
send() (AFL.automation.prepare.PrepareWidget.VBox (AFL.automation.APIServer.APIServer.Flask property), 104
method), 543 property), 104 send() (AFL.automation.prepare.SampleSeriesWidget.Buttsmend_protocol() (AFL.automation.prepare.Deck
method), 552 method), 370
send() (AFL.automation.prepare.SampleSeriesWidget.Cheskend_state() (AFL.automation.prepare.DeckBuilderWidget.Button
method), 560 method), 391
send() (AFL.automation.prepare.SampleSeriesWidget.FloasEnd_state() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
method), 570 method), 399
$send()$ (AFL.automation.prepare.SampleSeriesWidget.HBo $send_state()$ (AFL.automation.prepare.DeckBuilderWidget.Dropdown
method), 578 method), 412
send() (AFL.automation.prepare.SampleSeriesWidget.IntTesend_state() (AFL.automation.prepare.DeckBuilderWidget.HBox
method), 586 method), 420
send() (AFL.automation.prepare.SampleSeriesWidget.Labedend_state() (AFL.automation.prepare.DeckBuilderWidget.Label
method), 594 method), 427
$\verb send() (AFL. automation. prepare. Sample Series Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Layout \verb send() (AFL. automation. prepare. Deck Builder Widget. Deck$
method), 603 method), 437
$\verb send() (AFL. automation. prepare. Sample Series Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Text \verb send_state() (AFL. automation. prepare. Deck Builder Widget. Deck Builder Widg$
method), 614 method), 445
$\verb send() (AFL. automation. prepare. Sample Series Widget. VBo. \verb send_state() (AFL. automation. prepare. Deck Builder Widget. VBox Send_state()) (AFL. automation. Deck Builder Widget. Deck$
method), 622 method), 453
send() (AFL.automation.prepare.SweepBuilderWidget.Buttonation.prepare.PrepareWidget.Button
method), 634 method), 480
send() (AFL.automation.prepare.SweepBuilderWidget.Cheskendc_state() (AFL.automation.prepare.PrepareWidget.Checkbox
method), 642 method), 488
send() (AFL.automation.prepare.SweepBuilderWidget.HBosend_state() (AFL.automation.prepare.PrepareWidget.Dropdown method), 650 method), 497
send() (AFL.automation.prepare.SweepBuilderWidget.Labelend_state() (AFL.automation.prepare.PrepareWidget.HBox
method), 658 method), 505
send() (AFL.automation.prepare.SweepBuilderWidget.Layomend_state() (AFL.automation.prepare.PrepareWidget.Label
method), 667 method), 512
send() (AFL.automation.prepare.SweepBuilderWidget.Textsend_state() (AFL.automation.prepare.PrepareWidget.Layout
method), 677 method), 522
send() (AFL.automation.prepare.SweepBuilderWidget.VBosend_state() (AFL.automation.prepare.PrepareWidget.Text
method), 685 method), 535
send() (AFL.automation.shared.ServerDiscovery.Zeroconfsend_state() (AFL.automation.prepare.PrepareWidget.VBox
method), 778 method), 543
send_ld_plot() (AFL.automation.APIServer.APIServer.AB456d_vetate() (AFL.automation.prepare.SampleSeriesWidget.Button
method), 16, 92, 150 method), 552
send_array_as_jpg() send_state()(AFL.automation.prepare.SampleSeriesWidget.Checkbox
(AFL.automation.APIServer.APIServer method), 560
$method$), 16, 92, 150 $send_state()$ (AFL.automation.prepare.SampleSeriesWidget.FloatText)
send_command() (AFL.automation.loading.UltimusVPressureController
method), 56, 363, 364 send_state() (AFL.automation.prepare.SampleSeriesWidget.HBox
send_deck_cb() (AFL.automation.prepare.DeckBuilderWidget.DeckB uthbar)\ V5d&et

	<i>Vidget.IntText</i> 50, 329, 341		
method), 586	SensorPollingThread	(class	in
<pre>send_state() (AFL.automation.prepare.SampleSeriesV</pre>	Vidget.Label AFL.automation.	loading.LoadStopp	erDriver),
method), 594	241		
<pre>send_state() (AFL.automation.prepare.SampleSeriesV</pre>	<i>Vid§æn/søyæRø</i> llingThread	(class	in
method), 603	-	loading.SensorPoll	ingThread),
send_state()(AFL.automation.prepare.SampleSeriesV		O	77
method), 614	SerialCommsException,	79. 347. 781	
send_state() (AFL.automation.prepare.SampleSeriesV	- · · · · · · · · · · · · · · · · · · ·	(class	in
method), 622	•	loading.DummyPu	
send_state() (AFL.automation.prepare.SweepBuilderV			
method), 634	SerialDevice	(class	in
send_state() (AFL.automation.prepare.SweepBuilderV		,	
	viagei.Check bu st.automation 254	ioaaing.werksyrin	ger ump),
method), 642		(-1	÷
send_state() (AFL.automation.prepare.SweepBuilderV	_	(class	in
method), 650		loading.PressureCo	ontrollerAsPump),
send_state() (AFL.automation.prepare.SweepBuilder\)			
method), 658	SerialDevice	(class	in
send_state() (AFL.automation.prepare.SweepBuilder\)	• •	loading.SainSmarti	Relay),
method), 667	319		
$send_state()$ (AFL.automation.prepare.SweepBuilderV	_	(class	in
method), 677	AFL.automation.	loading.SerialDevi	ce), 53,
$\verb send_state() (AFL. automation. prepare. Sweep Builder Variable Sweep Builder Builder Sweep Builder Sweep Builder Builder Bu$	Widget.VBox 347		
method), 685	SerialDevice	(class	in
<pre>send_static_file() (AFL.automation.APIServer.APIS</pre>	Server.Flask AFL.automation	loading.ViciMultip	osSelector),
method), 122	365		
<pre>sendCommand() (AFL.automation.loading.ChemyxSyring</pre>	gePsanpi.alhiemey(x)Connection	(in	module
method), 33, 218, 222	AFL.automation.	shared.serializatioi	n), 80,
sendCommand() (AFL.automation.loading.DoubleViciM	ultiposSelect&ViciMultiposSe	elector	
method), 228	server (AFL.automation.A		er.ServiceInfo
sendCommand() (AFL.automation.loading.DummyPump			3
Senuconhana) (Ar L.amomanon.toaamy.Dummvrum)			
		hared ServerDisco	very.AsyncServiceInfo
method), 230	server (AFL.automation.s	hared.ServerDisco	very.AsyncServiceInfo
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe.	server (AFL.automation.s. Pump.SerialDoctriibate), 753		
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254	server (AFL.automation.s. Pump.SerialDetrübute), 753 server (AFL.automation.s.		
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringenethod), 254 sendCommand() (AFL.automation.loading.PressureCont	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumpt&abiadDeville	hared.ServerDisco	very.ServiceInfo
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringenethod), 254 sendCommand() (AFL.automation.loading.PressureContempthod), 293	server (AFL.automation.s. Pump.SerialDoctribute), 753 server (AFL.automation.s. trollerAsPumptSeibiadd)eViX& server_cmd() (AFL.auton	hared.ServerDisco nation.APIServer.C	very.ServiceInfo
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringen method), 254 sendCommand() (AFL.automation.loading.PressureCont method), 293 sendCommand() (AFL.automation.loading.SainSmartRei	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSeibiadDeViKe server_cmd() (AFL.autom lay.SainSmartRethoyd), 18, 153	hared.ServerDisco nation.APIServer.C , 155	very.ServiceInfo Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont method), 293 sendCommand() (AFL.automation.loading.SainSmartRel method), 319	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumpt&abiadDeVille server_cmd() (AFL.automaty.SainSmartRetluyd), 18, 153 server_cmd() (AFL.automaty.	hared.ServerDisco nation.APIServer.C , 155	very.ServiceInfo Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont method), 293 sendCommand() (AFL.automation.loading.SainSmartRei method), 319 sendCommand() (AFL.automation.loading.SainSmartRei	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptStibioalDevitCe server_cmd() (AFL.auton lay.SainSmartRetluyd), 18, 153 server_cmd() (AFL.auton lay.SerialDevivethod), 234	hared.ServerDisco nation.APIServer.C , 155 nation.loading.Loa	very.ServiceInfo Client.Client dStopperDriver.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRei. method), 319 sendCommand() (AFL.automation.loading.SainSmartRei. method), 319	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSabbadd)evit&e server_cmd() (AFL.autom lay.SainSmartikethoyd), 18, 153, server_cmd() (AFL.autom lay.SerialDevivethod), 234 server_cmd() (AFL.autom	hared.ServerDisco nation.APIServer.C , 155 nation.loading.Loa	very.ServiceInfo Client.Client dStopperDriver.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringel. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice.	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiadDeVitCe server_cmd() (AFL.automation.s. lay.SainSmartRetbuxd), 18, 153. server_cmd() (AFL.automation.s. lay.SerialDevivethod), 234 server_cmd() (AFL.automaticalDevivethod), 403	hared.ServerDisconation.APIServer.C , 155 nation.loading.Loanation.prepare.Dec	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiadA)eViCe server_cmd() (AFL.automation.s. trollerAsPumptSaibiadA)eViCe server_cmd() (AFL.automation.s. trollerAsPumptSaibiadA), 18, 153. server_cmd() (AFL.automaticalDevicethod), 234 server_cmd() (AFL.automaticalDevicemethod), 403 server_cmd() (AFL.automaticalDevicemethod), 403	hared.ServerDisconation.APIServer.C , 155 nation.loading.Loanation.prepare.Dec	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultiposS	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptStibiadDeVitCe server_cmd() (AFL.automation.s. trollerAsPumptStibiadDeVitCe server_cmd() (AFL.automation.s. server_cmd() (AFL.automaticalDevitCethod), 234 server_cmd() (AFL.automaticalDevitCethod), 403 server_cmd() (AFL.automaticalDevitCethod), 465	hared.ServerDisconation.APIServer.Constion.155 nation.loading.Loanation.prepare.Deconation.prepare.Deconation.prepare.OT.	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiaalDeviide server_cmd() (AFL.automation.s. trollerAsPumptSaibiaalDeviide server_cmd() (AFL.automation.s. server_cmd() (AFL.automation.s. SerialDevicemethod), 234 server_cmd() (AFL.automation.s. SerialDevicemethod), 403 server_cmd() (AFL.automation.s. S	hared.ServerDisconation.APIServer.Constion.155 nation.loading.Loanation.prepare.Deconation.prepare.Deconation.prepare.OT.	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost.	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSabiadDevit&e server_cmd() (AFL.automation.s. trollerAsPumptSabiadDevit&e server_cmd() (AFL.automation.s. trollerAsPumptSabiadDevit&e lay.SainSmartiRethood), 18, 153, server_cmd() (AFL.automation.serialDevitðod), 234 server_cmd() (AFL.automation.server_cmd() (AFL.auto	hared.ServerDisconation.APIServer.Co., 155 nation.loading.Loanation.prepare.Deconation.prepare.OT.	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringel. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSeibiadDevite server_cmd() (AFL.automation.s. trollerAsPumptSeibiadDevite server_cmd() (AFL.automation.s. server_cmd() (AFL.automation.s. SerialDevicemethod), 234 server_cmd() (AFL.automation.s. server_cmd() (AFL.automation.s	hared.ServerDisconation.APIServer.Co., 155 nation.loading.Loanation.prepare.Deconation.prepare.OT.	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost.	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiadDevice server_cmd() (AFL.automation.s. lay.SainSmartReduoxd), 18, 153, server_cmd() (AFL.automation.s. lay.SerialDevicethod), 234 server_cmd() (AFL.automation.s. SerialDevicemethod), 403 server_cmd() (AFL.automation.s. Selector.SerialDetrice, 465 server_cmd() (AFL.automation.s. Selector.ViciMeddipolsSelector server_cmd() (AFL.automation.s. Selector.ViciMeddipolsSelector server_cmd() (AFL.automation.s.)	hared.ServerDisconation.APIServer.Co., 155 mation.loading.Loanation.prepare.Deconation.prepare.OT. mation.prepare.OT. mation.prepare.OT. mation.prepare.San	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client npleSeriesWidget.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringel. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SainSmartRel. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSeibiadDevite server_cmd() (AFL.automation.s. trollerAsPumptSeibiadDevite server_cmd() (AFL.automation.s. server_cmd() (AFL.automation.s. SerialDevicemethod), 234 server_cmd() (AFL.automation.s. server_cmd() (AFL.automation.s	hared.ServerDisconation.APIServer.Co., 155 mation.loading.Loanation.prepare.Deconation.prepare.OT. mation.prepare.OT. mation.prepare.OT. mation.prepare.San	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client npleSeriesWidget.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366 Sensor (class in AFL.automation.loading.Sensor), 48	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiadA)eViCe server_cmd() (AFL.automation.s. lay.SainSmartRetluyd), 18, 153. server_cmd() (AFL.automation.s. lay.SerialDevicethod), 234 server_cmd() (AFL.automation.s. SerialDevicemethod), 403 server_cmd() (AFL.automation.s. Selector.SerialDetrice(), 465 server_cmd() (AFL.automation.s. Selector.ViciMudilipal)Selector server_cmd() (AFL.automation.s. method), 564 server_cmd() (AFL.automation.s.	hared.ServerDisconation.APIServer.Co., 155 mation.loading.Loanation.prepare.Deconation.prepare.OT. mation.prepare.OT. mation.prepare.San	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client npleSeriesWidget.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366 Sensor (class in AFL.automation.loading.Sensor), 48, 327	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptSaibiadA)eViCe server_cmd() (AFL.automation.s. lay.SainSmartRetluyd), 18, 153. server_cmd() (AFL.automation.s. lay.SerialDevicethod), 234 server_cmd() (AFL.automation.s. SerialDevicemethod), 403 server_cmd() (AFL.automation.s. Selector.SerialDetrice(), 465 server_cmd() (AFL.automation.s. Selector.ViciMudilipal)Selector server_cmd() (AFL.automation.s. method), 564 server_cmd() (AFL.automation.s.	hared.ServerDisconation.APIServer.Co., 155 mation.loading.Loanation.prepare.Deconation.prepare.OT. mation.prepare.OT. mation.prepare.San mation.prepare.San mation.sample.Cast	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client npleSeriesWidget.Client tingServer.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366 Sensor (class in AFL.automation.loading.Sensor), 48. 327 sensor_reset() (AFL.automation.loading.PneumaticP	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. rollerAsPumptSiibiadDeVitOe server_cmd() (AFL.automation.s. server_cmd() (AFL.automation), 18, 153. server_cmd() (AFL.automation), 234 server_cmd() (AFL.automation), 403 server_cmd() (AFL.automaticalDevicemethod), 403 server_cmd() (AFL.automaticalDevicemethod), 465 server_cmd() (AFL.automaticalDevicemethod), 564	hared.ServerDisconation.APIServer.Co., 155 nation.loading.Loanation.prepare.Deconation.prepare.OT. nation.prepare.OT. nation.prepare.San nation.prepare.San nation.sample.Cast PressureSampleCe nation.sample.Cast	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client npleSeriesWidget.Client tingServer.Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366 Sensor (class in AFL.automation.loading.Sensor), 48, 327 sensor_reset() (AFL.automation.loading.PneumaticP. method), 40, 277, 280	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. rollerAsPumptSiibiadDeVitOe server_cmd() (AFL.automation.s. server_cmd() (AFL.automation), 18, 153. server_cmd() (AFL.automation), 234 server_cmd() (AFL.automation), 403 server_cmd() (AFL.automaticalDevicemethod), 403 server_cmd() (AFL.automaticalDevicemethod), 465 server_cmd() (AFL.automaticalDevicemethod), 564	hared.ServerDisconation.APIServer.Co., 155 nation.loading.Loanation.prepare.Deconation.prepare.OT. nation.prepare.OT. nation.prepare.Sanation.prepare.Sanation.sample.CasteressureSample	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client ingServer.Client ll tingServer.OT2Client
method), 230 sendCommand() (AFL.automation.loading.NE1kSyringe. method), 254 sendCommand() (AFL.automation.loading.PressureCont. method), 293 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SainSmartRet. method), 319 sendCommand() (AFL.automation.loading.SerialDevice. method), 53, 347 sendCommand() (AFL.automation.loading.ViciMultipost. method), 365 sendCommand() (AFL.automation.loading.ViciMultipost. method), 366 Sensor (class in AFL.automation.loading.Sensor), 48, 327 sensor_reset() (AFL.automation.loading.PneumaticP. method), 40, 277, 280 sensor_reset() (AFL.automation.loading.PneumaticS.sensor_reset() (AFL.automation.loading.Pneu	server (AFL.automation.s. Pump.SerialDetribute), 753 server (AFL.automation.s. trollerAsPumptStibiotal)eVitte server_cmd() (AFL.automation.s. lay.SainSmartRetluyd), 18, 153. server_cmd() (AFL.automation.s. lay.SerialDevivethod), 234 server_cmd() (AFL.automatics. SerialDevicemethod), 403 server_cmd() (AFL.automatics. Selector.SerialDetrice, 465 server_cmd() (AFL.automatics. Selector.ViciModifipols)Selector server_cmd() (AFL.automatics. TressureSamplaclablePytetimatics. Server_cmd() (AFL.automatics. TressureSamplaclablePytetimatics. Server_cmd() (AFL.automatics. TressureSamplaclablePytetimatics. Server_key (AFL.automatics.	hared.ServerDisconation.APIServer.Co., 155 nation.loading.Loanation.prepare.Deconation.prepare.OT. nation.prepare.OT. nation.prepare.Sanation.prepare.Sanation.sample.CasteressureSample	very.ServiceInfo Client.Client dStopperDriver.Client ckBuilderWidget.Client 2Client.Client 2Client.OT2Client ingServer.Client ll tingServer.OT2Client

attribute), 753		method), 193
$\verb"server_key" (AFL. automation. shared. Server Discover automation and all all all all all all all all all al$	y.Ser	vs etIn @onfig()(AFL.automation.instrument.SeabreezeUVVis.Driver
attribute), 770		method), 195
ServerDiscovery (class	in	<pre>set_config() (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUV</pre>
AFL.automation.APIServer.Client), 154		method), 206
ServerDiscovery (class	in	<pre>set_config() (AFL.automation.instrument.SpecScreen_Driver.Driver</pre>
AFL.automation.shared.ServerDiscovery),		method), 209
78, 763, 779		$\verb"set_config" () (AFL. automation. instrument. Spec Screen_Driver. Spec Screen_Drive$
service_state_changed		method), 212
(AFL. automation. shared. Server Discovery. And the state of the sta	syncS	SesetceBookisg() (AFL.automation.loading.LoadStopperDriver.Client
attribute), 751		method), 235
service_state_changed		$\verb set_config() (AFL. automation. loading. Load Stopper Driver. Driver $
(AFL. automation. shared. Server Discovery. Set the substitute of the substitute o	ervice	
attribute), 768		$\verb set_config() (AFL. automation. loading. Load Stopper Driver. Load S$
ServiceBrowser (class	in	method), 241
AFL.automation.shared.ServerDiscovery),		$\verb set_config() (AFL. automation. loading. One Selector Blowout Sample Cell.$
764		method), 257
ServiceInfo (class AFL.automation.APIServer.APIServer), 139		<pre>set_config() (AFL.automation.loading.OneSelectorBlowoutSampleCell. method), 262</pre>
ServiceInfo (class	in	set_config()(AFL.automation.loading.OneSelectorBlowoutSampleCell.
AFL.automation.shared.ServerDiscovery),		method), 267
768		set_config()(AFL.automation.loading.PneumaticPressureSampleCell.L
ServiceStateChange (class	in	method), 271
AFL.automation.shared.ServerDiscovery),		$set_config()$ (AFL.automation.loading.PneumaticPressureSampleCell.F
773		method), 276
session_cookie_name		<pre>set_config() (AFL.automation.loading.PneumaticSampleCell.Driver</pre>
(AFL.automation.APIServer.APIServer.Flas	k	method), 282
property), 104		$\verb set_config() (AFL. automation. loading. Pneumatic Sample Cell. Pneumatic Sample Sam$
$\verb session_interface (AFL. automation. APIS erver. AFC) $	YSer	ver.Flask method), 286
attribute), 105		$\verb set_config() (AFL. automation. loading. PushPull Selector Sample Cell. Drawn and the property of the prop$
set_access_cookies() (in mod	lule	method), 295
AFL.automation.APIServer.APIServer), 88		$\verb set_config() (AFL. automation. loading. PushPull Selector Sample Cell. PushPull Selector$
<pre>set_aspirate_rate()</pre>		method), 300
	iver.L	D sætn<u>yc</u>MF2gD)i(AF L.automation.loading.RSoXSSolutionSampleCell.Drive
method), 62, 460, 462		method), 306
	ient	$\verb set_config() (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXSS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXS olutionS ample Cell. RSoXS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXS olutionS ample Cell. RSoXS of the config()) (AFL. automation. loading. RSoXS olutionS ample Cell. RSoXS olutionS olutionS ample Cell. RSoXS olutionS olutionS ample Cell. RSoXS olutionS olu$
method), 18, 153, 155		method), 312
	river	$set_config()$ (AFL.automation.loading.TwoSelectorBlowoutSampleCell.
method), 19, 158, 162		method), 351
- ·	river	:Betyeconfig() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.
method), 165		method), 358
- · ·	river	: Destruay Dfing() (AFL. automation. prepare. Deck Builder Widget. Client
method), 169)T) D	method), 403
- · ·	ועצוי	riset_Doirefig() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
method), 171	ハナンハ	method), 457
method), 175	'12D	ris et_Doonfriyf)(i)veA FL.automation.prepare.Dummy_OT2_Driver.Dummy_O method), 461
	IAC F	nemou), 401 Drivetr_config() (AFL.automation.prepare.OT2Client.Client
method), 184	110.D	method), 465
· · · · · · · · · · · · · · · · · · ·	SAST	nemou), 403 PasatuySchtfig() (AFL.automation.prepare.OT2Client.OT2Client
method), 187	11J.D	method), 469
	Driv	veret_config() (AFL.automation.prepare.SampleSeriesWidget.Client
method), 190		method), 564
	.1225	SANT_config() (AFL.automation.sample.CastingServer.CastingServer

method), 696	method), 286
$\mathtt{set_config()}\ (AFL. automation. sample. Casting Server. Cliese \verb t_dat $	
method), 698	method), 296
$set_config()$ (AFL.automation.sample.CastingServer.Drisert_dates)	· · · · · · · · · · · · · · · · · · ·
method), 700	method), 300
set_config()(AFL.automation.sample.CastingServer.OT 2 @tiedtat	
method), 705	method), 307
set_config() (AFL.automation.sample_env.TemperatureDeet_Ident	
method), 707	method), 312
set_config() (AFL.automation.sample_env.TemperatureDxet_Tlant	
method), 710	method), 352
set_data() (AFL.automation.APIServer.Driver.Driver set_dat	
method), 20, 159, 162	method), 358
set_data() (AFL.automation.APIServer.DummyDriver.Driset_dat	
method), 165	method), 457
set_data()(AFL.automation.APIServer.DummyDriver.DusentyDath	
method), 169	method), 461
set_data() (AFL.automation.APIServer.DummyOT2Driveseridat	
method), 172 set_data() (AFL.automation.APIServer.DummyOT2DrivesDurdaty	method), 696
method), 175	method), 701
<pre>set_data() (AFL.automation.instrument.DummySAS.Driveet_dat</pre>	method), 707
set_data()(AFL.automation.instrument.DummySAS.DumseySAGat	
method), 187	method), 710
set_data()(AFL.automation.instrument.I22SAXS.Driver set_dec	
method), 190	method), 60, 404, 455
set_data()(AFL.automation.instrument.I22SAXS.I22SAXSet_dec	
method), 193	method), 490
set_data()(AFL.automation.instrument.SeabreezeUVVis.Derbedis	
method), 196	(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_Dr
set_data() (AFL.automation.instrument.SeabreezeUVVis.Seabreeze	
	ver_object()
set_data()(AFL.automation.instrument.SpecScreen_Driver.Driver	
method), 210	method), 17, 92, 150
set_data()(AFL.automation.instrument.SpecScreen_Drivsesipensic	
method), 212	(AFL.automation.APIServer.Client.Client
set_data()(AFL.automation.loading.LoadStopperDriver.Driver	
	ver_object()
<pre>set_data() (AFL.automation.loading.LoadStopperDriver.LoadStop</pre>	
method), 241	method), 235
set_data()(AFL.automation.loading.OneSelectorBlowoutSetupdre	vid Dangect ()
method), 258	(AFL.automation.prepare.DeckBuilderWidget.Client
$\verb set_data() (AFL. automation. loading. One Selector Blowout Sample College Selector Blowout Select$	allethassel&0torBlowoutSampleCell
<pre>method), 262</pre>	ver_object()
$\verb set_data() (AFL. automation. loading. One Selector Blowout Sample College Selector Blowout Select$	Edl A F-V: aSetlenettotti Bhopræpt Sæn QT2Cd l ent. Client
method), 267	method), 465
set_data()(AFL.automation.loading.PneumaticPressureSsetpldCia	nlleDriobiject()
method), 272	(AFL.automation.prepare.OT2Client.OT2Client
$\verb set_data() (AFL. automation. loading. Pneumatic Pressure Sample Center Continuous C$	
method), 276 set_dri	ver_object()
$\verb set_data() (AFL. automation. loading. Pneumatic Sample Cell. Driver and the property of t$	· (AFL.automation.prepare.SampleSeriesWidget.Client
method), 282	method), 564
set_data()(AFL.automation.loading.PneumaticSampleCestePneumaticSam	meir Sabijde CE(I)

method), 235

method), 237

(AFL.automation.sample.CastingServer.Client method), 241
method), 699 set_object() (AFL.automation.loading.OneSelectorBlowoutSampleCell.
set_driver_object() method), 258
(AFL.automation.sample.CastingServer.OT2Clienset_object() (AFL.automation.loading.OneSelectorBlowoutSampleCell. method), 705 method), 262
set_gantry_speed() (AFL.automation.prepare.Dummy_ GE1_Divisec.D \textsigm(AFLQ\textsigm).loading.OneSelectorBlowoutSampleCell. method), 62, 460, 462 method), 267
set_mass() (AFL.automation.prepare.Component.Componert_object() (AFL.automation.loading.PneumaticPressureSampleCell.1
method), 59, 382, 384 method), 272
set_mass() (AFL.automation.prepare.factory.Solution set_object() (AFL.automation.loading.PneumaticPressureSampleCell.P
method), 690 method), 276
set_mass() (AFL.automation.prepare.Solute method), set_object() (AFL.automation.loading.PneumaticSampleCell.Driver
375 method), 282
set_mass() (AFL.automation.prepare.Solution set_object()(AFL.automation.loading.PneumaticSampleCell.Pneumati
method), 377 method), 286
set_mass() (AFL.automation.prepare.Solvent method), set_object() (AFL.automation.loading.PushPullSelectorSampleCell.Dr
380 method), 296
set_name() (AFL.automation.APIServer.APIServer.FileHandlerobject() (AFL.automation.loading.PushPullSelectorSampleCell.Pu
method), 97 method), 301
set_name() (AFL.automation.APIServer.APIServer.SMTP!sendlebject() (AFL.automation.loading.RSoXSSolutionSampleCell.Drive
method), 139 method), 307
set_object() (AFL.automation.APIServer.Client.Client set_object() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoX
method), 19, 153, 156 method), 312
set_object() (AFL.automation.APIServer.Driver set_object() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.
method), 20, 159, 162 method), 352
set_object() (AFL.automation.APIServer.DummyDriver.Detveobject() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.
method), 165 method), 358
set_object() (AFL.automation.APIServer.DummyDriver.DetruobDeixet) (AFL.automation.prepare.DeckBuilderWidget.Client
method), 169 method), 403
set_object() (AFL.automation.APIServer.DummyOT2Driset_Dibject() (AFL.automation.prepare.Dummy_OT2_Driver.Driver
method), 172 method), 457
set_object() (AFL.automation.APIServer.DummyOT2Driset_Dubject())v(AFL.automation.prepare.Dummy_OT2_Driver.Dummy_O
method), 175 method), 461
set_object() (AFL.automation.instrument.DummySAS.Drsetr_object() (AFL.automation.prepare.OT2Client.Client
method), 185 method), 465
set_object() (AFL.automation.instrument.DummySAS.DisentySdafect() (AFL.automation.prepare.OT2Client.OT2Client
method), 187 method), 469
set_object() (AFL.automation.instrument.I22SAXS.Driv&et_object() (AFL.automation.prepare.SampleSeriesWidget.Client
method), 190 method), 564
set_object() (AFL.automation.instrument.I22SAXS.I22SA\$\structure{S}_object() (AFL.automation.sample.CastingServer.CastingServer
method), 193 method), 696
set_object() (AFL.automation.instrument.SeabreezeUVViseDribrject() (AFL.automation.sample.CastingServer.Client
method), 196 method), 699
set_object() (AFL.automation.instrument.SeabreezeUVViseSeabriecatUVVAFL.automation.sample.CastingServer.Driver
method), 206 method), 701
set_object() (AFL.automation.instrument.SpecScreen_Dsietr.Dbiject() (AFL.automation.sample.CastingServer.OT2Client
method), 210 method), 705
set_object() (AFL.automation.instrument.SpecScreen_Dsixtr_Speceter_April:ventomation.sample_env.TemperatureDeck.Driver
method), 212 method), 707

882 Index

 $\verb|set_object(|)| (AFL. automation. loading. Load Stopper Driv \textit{set}(|))| (AFL. automation. sample_env. Temperature Deck. Temperature De$

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), 710

method), 720

method), 740 method), 267	
set_P() (AFL.automation.loading.DigitalOutPressureContsellersDigitalOutDressureContsellersDigitalOutDre	hattrollloading.PneumaticPressureSampleCell.
method), 34, 224, 225 method), 271	
set_P() (AFL.automation.loading.UltimusVPressureContrader_WhitesEllraCton	tuatler.loading.PneumaticPressureSampleCell
method), 56, 363, 364 method), 276	
set_params() (AFL.automation.shared.DataLabelerWidges.etr_dsamplreO)leAFL.autom	nation.loading.PneumaticSampleCell.Driver
method), 720 method), 282	
$\verb set_params() (AFL. automation. shared. Diffraction Labeler \verb Setdisealtplay (AFL. automation. shared. Diffraction Labeler automation. Diffraction Labele$	nation. loading. Pneumatic Sample Cell. Pneumat
method), 740 method), 286	
	$\it nation.loading. Push Pull Selector Sample Cell. Dr$
(AFL.automation.prepare.factory.Solution method), 295	
	$\it nation. loading. Push Pull Selector Sample Cell. Put$
set_properties_from_dict() method), 301	
	$\it nation. loading. RSo XSS olution Sample Cell. Drive$
377 method), 306	e i i na vaa i e i e ii na i
set_queue_mode() (AFL.automation.EpicsADLiveProcesssetiestamphen() (AFL.automation.EpicsADLiveProcessetiestamphen() (AFL.automation.EpicsADLiveProcessetiestamphen() (AFL.automation.EpicsADLiveProcessetiestamphen() (AFL.automation.EpicsADLiveProce	nation.toading.RSoXSSotutionSampleCett.RSo2
method), 24, 792, 793 method), 312 set_refresh_cookies() (in module set_sample() (AFL.autor	wation loading Two Salaston Planaut Samula Call
AFL.automation.APIServer.APIServer), 88 method), 351	$\it mation. loading. Two Selector Blowout Sample Cell$
set_sample() (AFL.automation.APIServer.Driver set_sample() (AFL.automation.APIServer.Driver set_sample() (AFL.automation.APIServer.Driver set_sample() (AFL.automation.APIServer.Driver.Driver set_sample() (AFL.automation.APIServer.Driver.Driver set_sample() (AFL.automation.APIServer.Driver.Driver set_sample() (AFL.automation.APIServer.Driver.Dr	nation loading TwoSelectorRlowoutSampleCell
method), 19, 158, 162 method), 358	nation.touaing.1woscicciorBiowoutsampicCett
set_sample() (AFL.automation.APIServer.DummyDriver.Betvesample() (AFL.auton	nation prepare Dummy OT2 Driver Driver
method), 165 method), 457	
set_sample() (AFL.automation.APIServer.DummyDriver.Betrusalipplief) (AFL.auton	nation.prepare.Dummy OT2 Driver.Dummy O
method), 169 method), 461	
set_sample() (AFL.automation.APIServer.DummyOT2Driset_Dsimple() (AFL.auton	nation.sample.CastingServer.CastingServer
method), 171 method), 696	
set_sample() (AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.APIServer.DummyOT2Driset_DamplyD())v(AFL.automation.DummyOT2Driset_DamplyD())v(AFL.automati	nation.sample.CastingServer.Driver
method), 175 method), 700	
set_sample() (AFL.automation.instrument.DummySAS.Drsetr_sample() (AFL.autor	nation.sample_env.TemperatureDeck.Driver
method), 184 method), 707	
set_sample()(AFL.automation.instrument.DummySAS.DusetruySample()(AFL.automation.instrument.DummySAS.DusetruySample()	nation.sample_env.TemperatureDeck.Temperat
method), 187 method), 710	
<pre>set_sample() (AFL.automation.instrument.I22SAXS.Driveset_sensor_config()</pre>	
	a.loading.PneumaticPressureSampleCell.Pneum
set_sample() (AFL.automation.instrument.I22SAXS.I22SAXS method), 40, 277	, 280
<pre>method), 193</pre>	loading Progratio Sample Cell Progratio Sam
method), 195 method), 42, 286	
set_sample() (AFL.automation.instrument.SeabreezeUVVie.SeabereeauUVVimissing	
	APIServer.APIServer.ServiceInfo
set_sample() (AFL.automation.instrument.SpecScreen_Driver.Drivemethod), 144	iserver iserver.servicetigo
method), 209 set_server_if_missing	1()
set_sample() (AFL.automation.instrument.SpecScreen_Driver.SpecSafeLnulDnivenion	
method), 212 method), 756	,
set_sample() (AFL.automation.loading.LoadStopperDrivesefDriserver_if_missing	J()
method), 236 (AFL.automation	.shared.ServerDiscovery.ServiceInfo
set_sample() (AFL.automation.loading.LoadStopperDriver.LoadStoppetnDdrive73	
method), 241 set_shake() (AFL.automa	ation.prepare.Dummy_OT2_Driver.Dummy_O
$\verb set_sample() (AFL. automation. loading. One Selector Blowout Sample \textit{Gethabliy} \textit{oft}, 460) \\$, 462
	L.automation.prepare.Dummy_OT2_Driver.Du
set_sample() (AFL.automation.loading.OneSelectorBlowoutSample@eth.Othe.SelectorBlowoutSample@eth.Othe.SelectorBlowoutSample.OneSelectorBlowoutSample.Othe.Se	
	ation.prepare.DeckBuilderWidget.Button
$\verb set_sample() (AFL. automation. loading. One Selector Blowout Sample \textit{Sell-eta}) \\ Q. \textit{Sell-eta} \\ loading. One Selector Blowout Sample Blowout Sam$	BlowoutSampleCell

- set_state() (AFL.automation.prepare.DeckBuilderWidgetsetbesklatte() (AFL.automation.prepare.SweepBuilderWidget.Label method), 399 method), 658
- set_state() (AFL.automation.prepare.DeckBuilderWidgesDtopdawn() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 412 method), 668
- ${\tt set_state()} \ (AFL. automation. prepare. Deck Builder Widgets \verb|WBostate()| (AFL. automation. prepare. Sweep Builder Widget. Textilization), 420 \\ method), 420 \\ method), 677$
- set_state() (AFL.automation.prepare.DeckBuilderWidgetsetate() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 427 method), 685
- set_state() (AFL.automation.prepare.DeckBuilderWidgetsEngeraget() (AFL.automation.prepare.MassBalance method), 437 method), 371
- set_state() (AFL.automation.prepare.DeckBuilderWidgesExttemp() (AFL.automation.prepare.Dummy_OT2_Driver.Dummy_OT2_method), 445 method), 61, 460, 462
 set_state() (AFL.automation.prepare.DeckBuilderWidgesVBotemp() (AFL.automation.sample_env.TemperatureDeck
- method), 453 method), 69, 709, 710 set_state() (AFL.automation.prepare.PrepareWidget.Button
- set_state() (AFL.automation.prepare.PrepareWidget.Butset_trait() (AFL.automation.prepare.DeckBuilderWidget.Button method), 480 method), 391
- ${\tt set_state()} \ (AFL. automation. prepare. Prepare Widget. Checkbox method), 488} \ method), 488 \ method), 399$
- set_state() (AFL.automation.prepare.PrepareWidget.Dro**get**owtrait() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 497 method), 412
- $\verb|set_state()| (AFL. automation. prepare. Prepare Widget. HBset_trait()| (AFL. automation. prepare. Deck Builder Widget. Deck Builder$
- set_state() (AFL.automation.prepare.PrepareWidget.Layset_trait() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 522 method), 437
- set_state() (AFL.automation.prepare.PrepareWidget.VB&et_trait() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 543
 method), 453
- set_state() (AFL.automation.prepare.SampleSeriesWidgeteCheckbox () (AFL.automation.prepare.PrepareWidget.Checkbox method), 560 method), 488
- set_state() (AFL.automation.prepare.SampleSeriesWidgeteFlortrEixt() (AFL.automation.prepare.PrepareWidget.Dropdown method), 570 method), 497
- set_state() (AFL.automation.prepare.SampleSeriesWidgetHBbrait() (AFL.automation.prepare.PrepareWidget.HBox method), 578 method), 505
- set_state() (AFL.automation.prepare.SampleSeriesWidgesehntTexait() (AFL.automation.prepare.PrepareWidget.Label method), 586 method), 512
- set_state() (AFL.automation.prepare.SampleSeriesWidgesetaltedait() (AFL.automation.prepare.PrepareWidget.Layout method), 594 method), 522
- set_state() (AFL.automation.prepare.SampleSeriesWidgesettaytemait() (AFL.automation.prepare.PrepareWidget.Text method), 604 method), 535
- set_state() (AFL.automation.prepare.SampleSeriesWidgetElextrait() (AFL.automation.prepare.PrepareWidget.VBox method), 614
 method), 543
- ${\tt set_state()}~(AFL. automation. prepare. Sample Series Widget {\tt Barait()}~(AFL. automation. prepare. Sample Series Widget. {\tt Burait()}~(AFL. automation. prepare. {\tt Burait()}~(AFL. a$
- set_state() (AFL.automation.prepare.SweepBuilderWidg**seButtva**it() (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 635 method), 560
- set_state() (AFL.automation.prepare.SweepBuilderWidg**seChtcHick**() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 642 method), 571
- set_state() (AFL.automation.prepare.SweepBuilderWidgseHBcmait() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 650 method), 578

```
set_trait() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 244
                                      method), 586
                                                                                                                                                                                                                                             setDaemon() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv)
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Label method), 247
                                                                                                                                                                                                                                             setDaemon() (AFL.automation.loading.LoadStopperDriver.StopLoadCBv2
                                       method), 594
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 250
                                                                                                                                                                                                                                             \mathtt{setDaemon()}\ (AFL. automation. loading. Sensor. Dummy Sensor 1
                                      method), 604
set_trait() (AFL.automation.prepare.SampleSeriesWidget.Text
                                                                                                                                                                                                                                                                                   method), 323
                                      method), 614
                                                                                                                                                                                                                                              setDaemon() (AFL.automation.loading.Sensor.DummySensor2
set_trait() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 326
                                      method), 622
                                                                                                                                                                                                                                             setDaemon() (AFL. automation. loading. Sensor Callback Thread. Sensor Callb
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Button method), 332
                                      method), 635
                                                                                                                                                                                                                                              setDaemon() (AFL. automation. loading. Sensor Callback Thread. Simple Thread.
\verb|set_trait()| (AFL. automation. prepare. Sweep Builder Widget. Checkboxethod), 334
                                      method), 643
                                                                                                                                                                                                                                             setDaemon() (AFL. automation. loading. Sensor Callback Thread. Stop Load Control of Callback Thread. Stop
set_trait() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 337
                                                                                                                                                                                                                                              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), 650
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Label method), 340
                                                                                                                                                                                                                                             setDaemon() (AFL.automation.loading.SensorPollingThread.SensorPollin
                                       method), 658
set_trait() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 346
                                      method), 668
                                                                                                                                                                                                                                              setDaemon() (AFL.automation.shared.ServerDiscovery.RunThread
set\_trait() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                                                                                                                                                                                  method), 762
                                                                                                                                                                                                                                             \mathtt{setDaemon()}\ (AFL. automation. shared. Server Discovery. Service Browser
                                      method), 677
set_trait() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 768
                                       method), 685
                                                                                                                                                                                                                                             setdefault() (AFL.automation.APIServer.Driver.PersistentConfig
set_units_cb() (AFL.automation.prepare.PrepareWidget.StockBuildwetMidget.62
                                      method), 528
                                                                                                                                                                                                                                             setdefault() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
set_units_cb() (AFL.automation.prepare.StockBuilderWidget.StockBathldd)Willget
                                      method), 66, 626, 628
                                                                                                                                                                                                                                             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), 269
                                      method), 59, 382, 384
                                                                                                                                                                                                                                             {\tt setdefault()} \ (AFL. automation. loading. Pneumatic Pressure Sample Cell. defeated and the pressure of t
set_volume() (AFL.automation.prepare.factory.Solution
                                                                                                                                                                                                                                                                                     method), 279
                                      method), 691
                                                                                                                                                                                                                                             setdefault() (AFL. automation. loading. Pneumatic Sample Cell. default disconnected by the set of the set o
set_volume()
                                                                                                   (AFL.automation.prepare.Solute
                                                                                                                                                                                                                                                                                     method), 288
                                       method), 375
                                                                                                                                                                                                                                             setdefault() (AFL. automation. loading. PushPullSelectorSampleCell. default()
                                                                                                                                                                                                                                                                                     method), 303
set_volume()
                                                                                          (AFL.automation.prepare.Solution
                                      method), 378
                                                                                                                                                                                                                                             setdefault() (AFL.automation.loading.RSoXSSolutionSampleCell.default
set_volume()
                                                                                              (AFL.automation.prepare.Solvent
                                                                                                                                                                                                                                                                                    method), 315
                                      method), 380
                                                                                                                                                                                                                                             setdefault() (AFL. automation. loading. Two Selector Blowout Sample Cell. a
setAllChannelsOff()
                                                                                                                                                                                                                                                                                     method), 360
                                      (AFL.automation.loading.SainSmartRelay.SainSmartRelay.Local (AFL.automation.shared.DatasetWidget.defaultdict
                                      method), 47, 319
                                                                                                                                                                                                                                                                                    method), 730
setChannels() (AFL.automation.loading.MultiChannelRetactMultiChannelRetactMultiChannels()
                                      method), 37, 251
                                                                                                                                                                                                                                                                                    method), 744
setChannels() (AFL.automation.loading.SainSmartRelay. SetMisGravite (Relay L.automation.shared.PersistentConfig.PersistentConfig.
                                      method), 318
                                                                                                                                                                                                                                                                                     method), 747
setChannels() (AFL.automation.loading.SainSmartRelay.SetDShlaytRelahFL.automation.loading.ChemyxSyringePump.ChemyxConn
                                      method), 47, 319
                                                                                                                                                                                                                                                                                    method), 33, 219, 223
```

setDaemon() (AFL.automation.loading.LoadStopperDriverS&xEirlPollimg(I)n(AdrL.automation.instrument.SeabreezeUVVis.SeabreezeU

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), 33, 219, 223

method), 27, 205, 207

method), 27, 205, 207

Index 885

method), 136

method), 181

method), 795

```
method), 27, 205, 207
                                                                                                                                                                                                                     method), 231
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), 27, 205, 207
                                                                                                                                                                                                                     method), 38, 253, 255
setFormatter() (AFL.automation.APIServer.APIServer.FiseMRatl&G) (AFL.automation.loading.NE1kSyringePump.SyringePump
                             method), 97
                                                                                                                                                                                                                     method), 255
setFormatter() (AFL.automation.APIServer.APIServer.SMFTRIAmedle (AFL.automation.loading.PressureControllerAsPump.Pressure
                                                                                                                                                                                                                     method), 43, 292, 294
                             method), 138
setLevel() (AFL.automation.APIServer.APIServer.FileHandtate() (AFL.automation.loading.PressureControllerAsPump.SyringeF
                              method), 97
                                                                                                                                                                                                                     method), 293
setLevel() (AFL.automation.APIServer.APIServer.SMTPIsertillate() (AFL.automation.loading.SyringePump.SyringePump
                              method), 138
                                                                                                                                                                                                                     method), 54, 348
setLevel() (AFL.automation.loading.ChemyxSyringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpscNringePumpsc
                             method), 32, 220, 222
                                                                                                                                                                                                                     method), 262
setName() (AFL.automation.APIServer.APIServer.QueueDsetRimseLevel() (AFL.automation.loading.OneSelectorBlowoutSampleC
                             method), 136
                                                                                                                                                                                                                     method), 266
setName() (AFL.automation.APIServer.QueueDaemon.QuesetRamselvevel() (AFL.automation.loading.PneumaticPressureSampleCe
                             method), 181
                                                                                                                                                                                                                     method), 40, 275, 280
setName() (AFL.automation.EpicsADLiveProcess.ReduceDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedeveeDarRimsRedveeDarRimsRedveeDa
                                                                                                                                                                                                                     method), 41, 285, 289
                             method), 795
setName() (AFL.automation.loading.LoadStopperDriver.ScstorPolialgeVall() (AFL.automation.loading.RSoXSSolutionSampleCell.R.
                             method), 244
                                                                                                                                                                                                                     method), 47, 311, 316
setName() (AFL.automation.loading.LoadStopperDriver.StoptRind&Budvel() (AFL.automation.loading.TwoSelectorBlowoutSampleC
                                                                                                                                                                                                                     method), 56, 357, 361
                              method), 247
setName() (AFL.automation.loading.LoadStopperDriver.StoptSauleStant()
                             method), 250
                                                                                                                                                                                                                     (AFL.automation.instrument.SeabreezeUVVis.SeabreezeUVVis
setName() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                     method), 27, 205, 207
                                                                                                                                                                                       \mathtt{setStream}() (AFL. automation. APIServer. APIServer. File Handler
                              method), 324
setName() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                     method), 97
                                                                                                                                                                                       setTime() (AFL.automation.loading.ChemyxSyringePump.ChemyxConnec
                             method), 326
setName() (AFL.automation.loading.SensorCallbackThread.SensorGaldhaakThreadl9, 223
                              method), 332
                                                                                                                                                                                       setUnits() (AFL.automation.loading.ChemyxSyringePump.ChemyxConn
setName() (AFL.automation.loading.SensorCallbackThread.SimpleThreatslood), GB, 219, 222
                             method), 334
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Button
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadh@Blodl), 391
                             method), 337
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Checkb
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadicaRv21), 399
                             method), 340
                                                                                                                                                                                       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), 346
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.HBox
setName() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                     method), 420
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Label
                             method), 763
setName() (AFL.automation.shared.ServerDiscovery.ServiceBrowsemethod), 427
                             method), 768
                                                                                                                                                                                      setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Layout
setParameter() (AFL.automation.loading.PushPullSelectorSample@edilh@dy\\P}ነብlSelectorSampleCell
                             method), 44, 299, 304
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.DeckBuilderWidget.Text
setRate() (AFL.automation.loading.ChemyxSyringePump.ChemyxComenleation.45
                              method), 33, 219, 223
                                                                                                                                                                                       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
                                                                                                                                                                                       \verb|setup_instance()| (AFL. automation. prepare. Prepare Widget. Button
                              method), 32, 220, 222
setRate() (AFL.automation.loading.ChemyxSyringePump.SyringePumphod), 480
                             method), 221
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.PrepareWidget.Checkbox
setRate() (AFL.automation.loading.DummyPump.DummyPump
                                                                                                                                                                                                                     method), 488
                              method), 35, 230, 231
                                                                                                                                                                                       setup_instance() (AFL.automation.prepare.PrepareWidget.Dropdown
```

method), 497

setRate() (AFL.automation.loading.DummyPump.SyringePump

```
setup_instance() (AFL.automation.prepare.PrepareWidsatMBsteLevel() (AFL.automation.loading.RSoXSSolutionSampleCell.R.
                                                                                               method), 47, 311, 316
             method), 505
setup_instance() (AFL.automation.prepare.PrepareWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidgetWidget
                                                                                               method), 56, 358, 361
             method), 512
setup_instance() (AFL.automation.prepare.PrepareWidghteldy_owntext_processor()
             method), 522
                                                                                               (AFL.automation.APIServer.APIServer.Flask
setup_instance() (AFL.automation.prepare.PrepareWidget.Text method), 114
             method), 535
                                                                                  shell_context_processors
setup_instance() (AFL.automation.prepare.PrepareWidget.VBox (AFL.automation.APIServer.APIServer.Flask
             method), 543
                                                                                               attribute), 106
setup_instance() (AFL.automation.prepare.SampleSeriesHodget_Regnore_error()
                                                                                               (AFL.automation.APIServer.APIServer.Flask
             method), 552
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.Checkbd), 116
                                                                                  shuffle()
             method), 560
                                                                                                          (AFL.automation.prepare.SampleSeries
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.FiboethRed), 374
             method), 571
                                                                                 SimpleThresholdCB
                                                                                                                                   (class
                                                                                                                                                              in
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.HANDL.automation.loading.SensorCallbackThread),
             method), 578
                                                                                               52, 332, 342
setup_instance() (AFL.automation.prepare.SampleSeries Widge(AHILexattomation.prepare.factory.Solution prop-
             method), 586
                                                                                               erty), 690
setup_instance() (AFL.automation.prepare.SampleSeries Widget Hubaltomation.prepare.Solution property), 377
             method), 594
                                                                                  sld
                                                                                            (AFL.automation.prepare.Component.Component
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.Lpmopuerty), 59, 382, 384
                                                                                 sld (AFL.automation.prepare.Solute property), 375
             method), 604
setup_instance() (AFL.automation.prepare.SampleSeries Widge H. Lautomation.prepare.Solvent property), 379
             method), 614
                                                                                 SMTPHandler
                                                                                                                              (class
setup_instance() (AFL.automation.prepare.SampleSeriesWidget.VABBL.automation.APIServer.APIServer), 136
                                                                                  Solute (AFL.automation.prepare.Component.PrepType
             method), 622
setup_instance() (AFL.automation.prepare.SweepBuilderWidget.Buttrohute), 382
             method), 635
                                                                                 Solute (AFL.automation.prepare.PrepType.PrepType at-
setup_instance() (AFL.automation.prepare.SweepBuilderWidget.Ghibaktleox63, 472, 473
             method), 643
                                                                                 Solute (class in AFL.automation.prepare), 374
setup_instance() (AFL.automation.prepare.SweepBuildestWidgesHBox (AFL.automation.prepare.factory.Solution
             method), 650
                                                                                               property), 690
setup_instance() (AFL.automation.prepare.SweepBuildestVindpesLautomation.prepare.Solution property),
             method), 658
setup_instance() (AFL.automation.prepare.SweepBuild&dViidgiphaAFIL.automation.prepare.Component.PrepType
             method), 668
                                                                                               attribute), 382
setup_instance() (AFL.automation.prepare.SweepBuild&WWdgioTe(AFL.automation.prepare.PrepType.PrepType
                                                                                               attribute), 63, 472, 473
             method), 677
setup_instance() (AFL.automation.prepare.SweepBuild&dVindgioVIBobass in AFL.automation.prepare), 376
             method), 685
                                                                                 Solution (class in AFL.automation.prepare.factory),
setupDefaults() (AFL.automation.APIServer.QueueDaemon.DataTbashcan
             method), 179
                                                                                 Solvent (AFL.automation.prepare.Component.PrepType
setVolume() (AFL.automation.loading.ChemyxSyringePump.ChemyaCribmae);082
             method), 33, 219, 223
                                                                                 Solvent (AFL.automation.prepare.PrepType.PrepType
setWasteLevel() (AFL.automation.loading.OneSelectorBlowoutSamphib.ede), OneSelectorBlowoutSamphib.ede), OneSelectorBlowoutSamphib.ede)
                                                                                 Solvent (class in AFL.automation.prepare), 378
             method), 262
setWasteLevel() (AFL.automation.loading.OneSelectorBkolvnetStundelaSklautonBltwonptSapapleGelbry.Solution
             method), 267
                                                                                               property), 691
setWasteLevel() (AFL.automation.loading.PneumaticPresolveSaturgeleGslitPyneumAdIc.RutosmurtiSompregatel.Solution
             method), 40, 275, 280
                                                                                               property), 378
setWasteLevel() (AFL.automation.loading.PneumaticSampleCedtPnassutAiFSampleCedtlon.prepare.factory.Solution
             method), 41, 285, 289
                                                                                               property), 691
```

- solvent_mass (AFL.automation.prepare.Solution property), 378
- solvent_sld (AFL.automation.prepare.factory.Solution property), 691
- solvent_sld (AFL.automation.prepare.Solution property), 378
- $\verb"solvent_volume" (AFL. automation. prepare. factory. Solution$ property), 691 start() (AFL.automation.loading.SensorCallbackThread.SensorCallback
- solvent_volume (AFL.automation.prepare.Solution property), 378
- solvents (AFL.automation.prepare.factory.Solution property), 690
- solvents (AFL.automation.prepare.Solution property), 377
- SpecScreen_Driver (class in AFL.automation.instrument.SpecScreen_Driver), start() (AFL.automation.loading.SensorPollingThread.SensorPo
- 28, 210, 212 method), 346
- split_vars() (AFL.automation.shared.DatasetWidget.DatsstatWidgetAMLoahelomation.prepare.DeckBuilderWidget.DeckBuilderWidget method), 73, 726, 731 method), 60, 404, 455

method), 247

method), 250

method), 324

method), 326

method), 332

method), 334

method), 337

method), 340

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 method), 60, 405, 455
- $\verb|sqrt()| (in module AFL. automation. APIS erver. Dummy Driver)| (AFL. automation. prepare . Prepare Widget. Deck Builder Widget . Deck Builder . Deck Buil$ *method*), 491
- sqrt() (in module AFL.automation.APIServer.DummyOT2Btimet)() (AFL.automation.prepare.PrepareWidget.PrepareWidget method), 64, 524, 545
- sqrt() (in module AFL.automation.instrument.SpecScreen_\$Dairtr()) (AFL.automation.prepare.PrepareWidget.PrepareWidget_View method), 64, 525, 545
- sqrt() (in module AFL.automation.prepare.DeckBuilderWidget); () (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget method), 527
- $\mathtt{sqrt}()$ (in module AFL.automation.prepare.PrepareWidgetStart() (AFL.automation.prepare.PrepareWidget.StockBuilderWidgetStart()) method), 528
- sqrt() (in module AFL.automation.prepare.SampleSeriesWidger),() (AFL.automation.prepare.PrepareWidget.SweepBuilderWidget method), 529
- sqrt() (in module AFL.automation.prepare.StockBuilderWistgen); () (AFL.automation.prepare.SampleSeriesWidget method), 65, 607, 624
- sqrt() (in module AFL.automation.prepare.SweepBuilderWsitleart,() (AFL.automation.prepare.SampleSeriesWidget method), 66, 608, 625
- sqrt() (in module AFL.automation.sample.CastingServer), start() (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget. method), 67, 626, 628
- sqrt() (in module AFL.automation.shared.DataLabelerWidgetStrt() (AFL.automation.prepare.StockBuilderWidget.StockBuilderWidget. method), 67, 627, 628
- sqrt() (in module AFL.automation.shared.DiffractionLabekstart() (AFL.automation.prepare.SweepBuilderWidget.SweepBuilderwoopBuilderWidget.SweepBuilderWidget.SweepBuilderwoopBuilderwoopB method), 68, 670, 687 start() (AFL.automation.APIServer.APIServer.QueueDaestowrt() (AFL.automation.prepare.SweepBuilderWidget.Swee
- method), 136 method), 68, 671, 687 start() (AFL.automation.APIServer.APIServer.Zeroconf start() (AFL.automation.shared.ServerDiscovery.RunThread
- *method*), 146 method), 763
- start() (AFL.automation.APIServer.QueueDaemon.QueueDaemoh() (AFL.automation.shared.ServerDiscovery.ServiceBrowser method), 768 *method*), 182
- start() (AFL.automation.EpicsADLiveProcess.ReduceDaestant.ReduceDa method), 795 method), 776
- start() (AFL.automation.loading.LoadStopperDriver.SensotRanland(AFLautomation.APIServer.APIServer.Zeroconf method), 244 property), 146
- start() (AFL.automation.loading.LoadStopperDriver.Stop&badCeB (AFL.automation.shared.ServerDiscovery.Zeroconf

```
property), 776
                                                                                                                                                                                                                                 method), 41, 285, 289
startPump() (AFL.automation.loading.ChemyxSyringePunspathesn()x(Childrentionnation.loading.PushPullSelectorSampleCell.Driver
                               method), 33, 219, 222
                                                                                                                                                                                                                                 method), 296
stat() (AFL.automation.instrument.SeabreezeUVVis.Path status() (AFL.automation.loading.PushPullSelectorSampleCell.PushPull
                               method), 200
                                                                                                                                                                                                                                  method), 44, 299, 304
static_folder(AFL.automation.APIServer.APIServer.Flastatus() (AFL.automation.loading.RSoXSSolutionSampleCell.Driver
                               property), 122
                                                                                                                                                                                                                                  method), 307
static_url_path(AFL.automation.APIServer.APIServer.Etwatus()(AFL.automation.loading.RSoXSSolutionSampleCell.RSoXSSolu
                               property), 122
                                                                                                                                                                                                                                  method), 46, 310, 315
status()
                                                (AFL.automation.APIServer.Driver.Driver status() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.Drive
                               method), 19, 159, 162
                                                                                                                                                                                                                                  method), 352
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), 165
                                                                                                                                                                                                                                  method), 55, 356, 361
status() (AFL.automation.APIServer.DummyDriver.Dummy\text{\text{Driver}}() (AFL.automation.prepare.Dummy\text{\text{Driver}}.Driver\text{Driver})
                                method), 21, 168, 169
                                                                                                                                                                                                                                  method), 457
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), 172
                                                                                                                                                                                                                                  method), 61, 460, 462
status() (AFL.automation.APIServer.DummyOT2Driver.DatawayDf)vAFL.automation.sample.CastingServer.CastingServer
                               method), 21, 174, 175
                                                                                                                                                                                                                                  method), 82, 695, 705
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), 24, 791
                                                                                                                                                                                                                                  method), 701
status() (AFL.automation.instrument.DummySAS.Driver status() (AFL.automation.sample_env.TemperatureDeck.Driver
                                method), 185
                                                                                                                                                                                                                                  method), 707
status() (AFL.automation.instrument.DummySAS.DummyS4Stus() (AFL.automation.sample_env.TemperatureDeck.TemperatureDeck
                               method), 26, 186, 188
                                                                                                                                                                                                                                 method), 69, 709, 710
status() (AFL.automation.instrument.I22SAXS.Driver stem (AFL.automation.instrument.SeabreezeUVVis.Path
                                                                                                                                                                                                                                 property), 203
                                method), 190
status() (AFL.automation.instrument.I22SAXS.I22SAXS step (AFL.automation.prepare.SampleSeriesWidget.FloatText
                                                                                                                                                                                                                                  attribute), 567
                               method), 193
status() (AFL.automation.instrument.SeabreezeUVVis.Dristeep (AFL.automation.prepare.SampleSeriesWidget.IntText
                                method), 196
                                                                                                                                                                                                                                   attribute), 582
status()(AFL.automation.instrument.SeabreezeUVVis.Se&brackBUiVider_reset_cb()
                               method), 206
                                                                                                                                                                                                                                  (AFL.automation.prepare.PrepareWidget.PrepareWidget
                                                                                                                                                                                                                                 method), 64, 524, 545
status() (AFL.automation.instrument.SpecScreen_Driver.Driver
                                method), 210
                                                                                                                                                                                                 StockBuilderWidget
                                                                                                                                                                                                                                                                                                                        (class
                                                                                                                                                                                                                                                                                                                                                                                     in
status() (AFL.automation.instrument.SpecScreen_Driver.SpecScreeAFDnivetomation.prepare.PrepareWidget),
                               method), 28, 211, 212
status()(AFL.automation.loading.LoadStopperDriver.Dr&tockBuilderWidget
                                                                                                                                                                                                                                                                                                                        (class
                                                                                                                                                                                                                                                                                                                                                                                     in
                                                                                                                                                                                                                                 AFL.automation.prepare.StockBuilderWidget),
                                method), 237
status() (AFL.automation.loading.LoadStopperDriver.LoadStopperDriv@25, 627
                                                                                                                                                                                                 StockBuilderWidget_Model
                               method), 36, 240, 250
                                                                                                                                                                                                                                                                                                                                     (class
                                                                                                                                                                                                                                                                                                                                                                                     in
status() (AFL.automation.loading.OneSelectorBlowoutSampleCell.PFikarutomation.prepare.StockBuilderWidget),
                               method), 258
                                                                                                                                                                                                                                  67, 626, 628
status() (AFL.automation.loading.OneSelectorBlowoutSastpletBilliOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOneStalicOn
                                                                                                                                                                                                                                                                                                                                                                                     in
                               method), 263
                                                                                                                                                                                                                                 AFL.automation.prepare.StockBuilderWidget),
status() (AFL.automation.loading.OneSelectorBlowoutSampleCell.\textbf{WpSelectionBlowoutSampleCell}
                                method), 265
                                                                                                                                                                                                 stop() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
status() (AFL.automation.loading.PneumaticPressureSampleCell.Dniednod), 32, 220, 222
                                method), 272
                                                                                                                                                                                                  stop() (AFL.automation.loading.ChemyxSyringePump.SyringePump
status() (AFL.automation.loading.PneumaticPressureSampleCell.Pmeuhovat); PressureSampleCell
                               method), 40, 275, 279
                                                                                                                                                                                                  stop() (AFL. automation. loading. Digital Out Pressure Controller. Digital Out Pressure Controlle
status() (AFL.automation.loading.PneumaticSampleCell.Driver
                                                                                                                                                                                                                            method), 224
```

status() (AFL.automation.loading.PneumaticSampleCell.PneumatiosampleCell5

stop() (AFL.automation.loading.DigitalOutPressureController.PressureC

method), 282

attribute), 441

```
stop() (AFL.automation.loading.DummyPump.DummyPumpyPumpyPum (AFL.automation.prepare.PrepareWidget.Button
                                  method), 35, 230, 231
                                                                                                                                                                                                                                                attribute), 476
stop() (AFL.automation.loading.DummyPump.SyringePunstyle (AFL.automation.prepare.PrepareWidget.Checkbox
                                  method), 231
                                                                                                                                                                                                                                                attribute), 484
stop() (AFL:automation.loading.NE1kSyringePump.NE1kSyringe(Puffipautomation.prepare.PrepareWidget.Dropdown
                                 method), 37, 253, 255
                                                                                                                                                                                                                                               attribute), 497
stop() (AFL.automation.loading.NE1kSyringePump.SyringerPythep
                                                                                                                                                                                                                                               (AFL.automation.prepare.PrepareWidget.Label
                                  method), 254
                                                                                                                                                                                                                                                attribute), 508
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), 42, 290, 291
                                                                                                                                                                                                                                                tribute), 531
stop() (AFL.automation.loading.PressureControllerAsPungt.PressAreContoollatieAssPonepoure.SampleSeriesWidget.Button
                                  method), 43, 292, 294
                                                                                                                                                                                                                                                attribute), 548
stop() (AFL:automation.loading.PressureControllerAsPunst$\sinAdHunytomation.prepare.SampleSeriesWidget.Checkbox
                                  method), 293
                                                                                                                                                                                                                                                attribute), 556
\verb|stop()| (AFL. automation. loading. Syringe Pump. Syringe Pump\texttt{xyle} (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Float Text) | Stop() (AFL. automation. prepare. Sample Series Widget. Sample Series Widget. Prepare. Sample Series Widget. Sample Series
                                  method), 54, 348
                                                                                                                                                                                                                                                attribute), 571
stop() (AFL.automation.loading.UltimusVPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlstayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureControlStayPressureCont
                                  method), 362
                                                                                                                                                                                                                                               attribute), 586
stop() (AFL.automation.loading.UltimusVPressureControlsaryWhithAFLYMhrassuntiGupmaplure.SampleSeriesWidget.Label
                                 method), 363
                                                                                                                                                                                                                                                attribute), 590
stop_shake() (AFL.automation.prepare.Dummy_OT2_Drivey.DataFils_.Off@mDatione.prepare.SampleSeriesWidget.Text
                                 method), 61, 460, 462
                                                                                                                                                                                                                                               attribute), 610
stopLoad() (AFL.automation.loading.PneumaticPressureSstrypleC&ITLP.nationnaticPressureSatisplecGBlitlderWidget.Button
                                  method), 40, 275, 280
                                                                                                                                                                                                                                                attribute), 631
stopLoad() (AFL:automation.loading.PneumaticSampleCest.Pheu(AhFliaSutoupketGent)prepare.SweepBuilderWidget.Checkbox
                                  method), 41, 285, 289
                                                                                                                                                                                                                                               attribute), 639
stopLoad() (AFL.automation.loading.SensorCallbackThrewithDau/dkFCanthonmatiatincpurepare.SweepBuilderWidget.Label
                                 method), 52, 329, 343
                                                                                                                                                                                                                                                attribute), 654
StopLoadCBv1
                                                                                                                (class
                                                                                                                                                                                                            style (AFL.automation.prepare.SweepBuilderWidget.Text
                                 AFL.automation.loading.LoadStopperDriver),
                                                                                                                                                                                                                                                attribute), 673
                                  244
                                                                                                                                                                                                             submit_cb() (AFL.automation.prepare.PrepareWidget.SampleSeriesWidg
StopLoadCBv1
                                                                                                                (class
                                                                                                                                                                                              in
                                                                                                                                                                                                                                               method), 526
                                 AFL.automation.loading.SensorCallbackThread), submit_cb() (AFL.automation.prepare.SampleSeriesWidget.SampleSeries
                                  50, 335, 341
                                                                                                                                                                                                                                               method), 65, 606, 624
StopLoadCBv2
                                                                                                                (class
                                                                                                                                                                                                             submit_mixing_wells_cb()
                                 AFL.automation.loading.LoadStopperDriver),
                                                                                                                                                                                                                                               (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget
                                  247
                                                                                                                                                                                                                                               method), 527
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
                                  51, 338, 342
                                                                                                                                                                                                                                               method), 65, 607, 624
stopPump() (AFL.automation.loading.ChemyxSyringePumpx(Hring)(AGhmeattionation.instrument.SeabreezeUVVis.Path
                                 method), 33, 219, 222
                                                                                                                                                                                                                                               property), 203
                                                                                                                                                                                                            {\tt suffixes}\, (AFL. automation. instrument. Seabreeze UVV is. Path
strtobool()
                                                                                                         (in
                                                                                                                                                                           module
                                 AFL.automation.APIServer.APIServer), 89
                                                                                                                                                                                                                                               property), 203
\verb|style| (AFL. automation.prepare.DeckBuilderWidget.ButtonSweepBuilder\_reset\_cb()|
                                                                                                                                                                                                                                                (AFL.automation.prepare.PrepareWidget.PrepareWidget
                                  attribute), 387
                                                                                                                                                                                                                                               method), 64, 524, 545
style(AFL.automation.prepare.DeckBuilderWidget.Checkbox)
                                  attribute), 395
                                                                                                                                                                                                             SweepBuilderWidget
                                                                                                                                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                                                                                                                                           in
style(AFL. automation. prepare. Deck Builder Widget. Dropdown
                                                                                                                                                                                                                                               AFL.automation.prepare.PrepareWidget),
                                  attribute), 412
\verb|style| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|SweepBuilder Widget|)| | SweepBuilder Widget | SweepBuilder Widget | SweepBuilder Widget| | SweepBuilder Widget | SweepBuilder Widget| | 
                                                                                                                                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                                                                                                                                           in
                                                                                                                                                                                                                                               AFL.automation.prepare.SweepBuilderWidget),
                                 attribute), 423
{\tt style}\,(AFL. automation. prepare. Deck Builder Widget. Text
                                                                                                                                                                                                                                                67, 669, 687
```

890 Index

SweepBuilderWidget_Model

(class

in

```
AFL.automation.prepare.SweepBuilderWidget), tabbable (AFL.automation.prepare.PrepareWidget.Button
             68, 670, 687
                                                                                                  attribute), 480
SweepBuilderWidget_View
                                                                                    tabbable (AFL.automation.prepare.PrepareWidget.Checkbox
             AFL. automation. prepare. Sweep Builder Widget),
                                                                                                  attribute), 488
              68, 671, 687
                                                                                    tabbable (AFL.automation.prepare.PrepareWidget.Dropdown
swish() (AFL.automation.loading.OneSelectorBlowoutSampleCell.OntaBidlutatorBlowoutSampleCell
              method), 263
                                                                                    tabbable (AFL.automation.prepare.PrepareWidget.HBox
swish() (AFL.automation.loading.OneSelectorBlowoutSampleCell.TwttSiHutt@rBlowoutSampleCell
              method), 266
                                                                                    tabbable (AFL.automation.prepare.PrepareWidget.Label
swish() (AFL.automation.loading.PushPullSelectorSampleCell.PushAttuliSule);t&tSampleCell
              method), 45, 301, 305
                                                                                    tabbable (AFL.automation.prepare.PrepareWidget.Text
swish() (AFL.automation.loading.RSoXSSolutionSampleCell.RSoXS&ahibioneSampleCell
             method), 46, 311, 316
                                                                                    tabbable (AFL. automation. prepare. Prepare Widget. VBox
swish() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.TwttSithateorBlowoutSampleCell
              method), 56, 357, 361
                                                                                    {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. Button
symlink_to() (AFL.automation.instrument.SeabreezeUVVis.Path attribute), 552
                                                                                    {\tt tabbable} (AFL. automation. prepare. Sample Series Widget. Checkbox
             method), 201
sync_to_prepare_cb()
                                                                                                  attribute), 560
              (AFL.automation.prepare.PrepareWidget.SampleSexteds MilleAtAFL.automation.prepare.SampleSeriesWidget.FloatText
             method), 527
                                                                                                  attribute), 571
                                                                                    {\tt tabbable} \, (AFL. automation. prepare. Sample Series Widget. HBox
sync_to_prepare_cb()
              (AFL.automation.prepare.SampleSeriesWidget.SampleSeriesWidget), 578
             method), 65, 607, 624
                                                                                    {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. Int Text
SyringePump
                                             (class
                                                                                                  attribute), 586
             221
                                                                                                  attribute), 594
SyringePump
                                             (class
                                                                                    {\tt tabbable} \ (AFL. automation. prepare. Sample Series Widget. Text
             AFL.automation.loading.DummyPump),
                                                                                                  attribute), 614
                                                                                    tabbable (AFL.automation.prepare.SampleSeriesWidget.VBox
SyringePump
                                             (class
                                                                             in
                                                                                                  attribute), 622
             AFL.automation.loading.NE1kSyringePump),
                                                                                    tabbable (AFL.automation.prepare.SweepBuilderWidget.Button
              254
                                                                                                  attribute), 635
                                                                                   tabbable (AFL.automation.prepare.SweepBuilderWidget.Checkbox
SyringePump
                                             (class
                                                                              in
             AFL.automation.loading.PressureControllerAsPump),
                                                                                                  attribute), 643
              293
                                                                                    tabbable (AFL.automation.prepare.SweepBuilderWidget.HBox
SyringePump
                                             (class
                                                                                                  attribute), 650
                                                                             in
             AFL.automation.loading.SyringePump),
                                                                            53,
                                                                                    tabbable (AFL.automation.prepare.SweepBuilderWidget.Label
              348
                                                                                                  attribute), 658
                                                                                    tabbable (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                  attribute), 678
{\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), 685
              attribute), 391
tabbable (AFL.automation.prepare.DeckBuilderWidget.Chrakloot (AFL.automation.prepare.Sample property), 373
                                                                                    target_check (AFL.automation.prepare.Sample prop-
              attribute), 399
tabbable (AFL.automation.prepare.DeckBuilderWidget.Dropdown erty), 373
                                                                                    target_loc (AFL.automation.prepare.Sample prop-
              attribute), 412
tabbable (AFL.automation.prepare.DeckBuilderWidget.HBox
                                                                                                  erty), 373
                                                                                    teardown_appcontext()
              attribute), 420
{\tt tabbable} \ (AFL. automation. prepare. Deck Builder Widget. Label
                                                                                                  (AFL.automation.APIServer.APIServer.Flask
                                                                                                  method), 114
              attribute), 428
{\tt tabbable} \ (AFL. automation. prepare. Deck Builder Widget. \textit{Text} \textbf{mear} down\_app \textbf{context\_funcs}
                                                                                                  (AFL.automation.APIServer.APIServer.Flask
              attribute), 445
                                                                                                  attribute), 106
tabbable (AFL.automation.prepare.DeckBuilderWidget.VBox
                                                                                    teardown_request()(AFL.automation.APIServer.APIServer.Flask
              attribute), 453
```

method), 122	method), 111
teardown_request_funcs	test_cli_runner_class
(AFL.automation.APIServer.APIServer.Flask	(AFL.automation.APIServer.APIServer.Flask
attribute), 124	attribute), 105
TemperatureDeck (class in	${\sf test_client()}$ (AFL.automation.APIServer.APIServer.Flask
$AFL. automation. sample_env. Temperature Deck),\\$	method), 110
69, 708, 710	${\tt test_client_class} (AFL. automation. APIS erver. APIS erver. Flask$
template_context_processors	attribute), 105
(AFL. automation. APIS erver. APIS erver. Flask	${\sf test_command1()}\ (AFL. automation. APIS erver. Dummy Driver. Dummy$
attribute), 124	method), 21, 168, 169
template_filter() (AFL.automation.APIServer.APISer method), 113	vetreElasscommand1() (AFL.automation.APIServer.DummyOT2Driver.Dummy method), 22, 174, 175
template_folder(AFL.automation.APIServer.APIServer	r.Flast_command2()(AFL.automation.APIServer.DummyDriver.DummyDriv
attribute), 123	method), 21, 168, 169
<pre>template_global() (AFL.automation.APIServer.APISer</pre>	vtce Etas c command 2 () (AFL. automation. APIS erver. Dummy OT2 Driver. Dummy
method), 113	method), 22, 174, 176
<pre>template_test() (AFL.automation.APIServer.APIServer</pre>	:Fest_command_sets_data()
method), 113	(AFL. automation. API Server. Dummy Driver. Dummy Driver
templates_auto_reload	method), 21, 168, 169
(AFL. automation. APIS erver. APIS erver. Flask	test_command_sets_data()
property), 108	(AFL. automation. API Server. Dummy OT 2 Driver. Dummy Driver
${\tt terminate()} \ (AFL. automation. APIS erver. APIS erver. Quentum approximation and approximation approximation and approximation approximation and approximation and approximation and approximation and approximation approximation approximation and approximation approximation approximation approximation approximation and approximation approximat$	
method), 135	test_image()(AFL.automation.APIServer.DummyDriver.DummyDriver
terminate() (AFL.automation.APIServer.QueueDaemon.	
method), 23, 180, 182	test_image()(AFL.automation.APIServer.DummyOT2Driver.DummyDri
terminate() (AFL.automation.EpicsADLiveProcess.Redu	
method), 25, 794, 796	test_plot() (AFL.automation.APIServer.DummyDriver.DummyDriver
terminate()(AFL.automation.loading.LoadStopperDrive	
method), 243	test_plot() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver.D
terminate()(AFL.automation.loading.LoadStopperDrive	
method), 247	test_request_context()
terminate() (AFL.automation.loading.LoadStopperDrive method), 250	method), 126
terminate() (AFL.automation.loading.Sensor.DummySen	
method), 49, 322, 327	attribute), 104
${\tt terminate()} \ (AFL. automation. loading. Sensor. Dummy Sensor. Dum$	
method), 49, 325, 328	attribute), 141
terminate() (AFL.automation.loading.SensorCallbackTh method), 50, 330, 341	aread:SeAFdr.Galbbaakibhrdudred.ServerDiscovery.AsyncServiceInfo attribute), 753
	aread:S(AuFle.Tihteshalid6.Bhared.ServerDiscovery.ServiceInfo
method), 334	attribute), 770
${\tt terminate()} \ (AFL. automation. loading. Sensor Callback Theorem 1991) \ (AFL. automation. loading. Sensor Callback Theorem 2991) \ (AFL. automation. lo$	nte art:StopIsoandABL lautomation.prepare.DeckBuilderWidget),
method), 337	439
terminate() (AFL.automation.loading.SensorCallbackTh method), 340	529
terminate() (AFL.automation.loading.SensorPollingThromethod), 53, 345, 346	e T.&Str(sbxRoib li AFII)aead mation.prepare.SampleSeriesWidget), 608
	alidker(class in AFL.automation.prepare.SweepBuilderWidget),
attribute), 97	671
ternary_click_callback()	timed_dispense() (AFL.automation.loading.DigitalOutPressureControll
(AFL.automation.shared.DiffractionLabeler.Diffr	
method), 75, 733, 741	timed_dispense() (AFL.automation.loading.DigitalOutPressureControl
test() (in module AFL.automation), 13, 789	method), 225
	vtrifiktskdispense() (AFL.automation.loading.PressureController.Pressure

method), 42, 290 tooltip (AFL.automation.prepare.PrepareWidget.Text
timed_dispense() (AFL.automation.loading.UltimusVPressureController
method), 362 tooltip (AFL.automation.prepare.PrepareWidget.VBox
timed_dispense() (AFL.automation.loading.UltimusVPressureController
method), 363 tooltip (AFL.automation.prepare.SampleSeriesWidget.Button
to_dict() (AFL.automation.prepare.factory.Solution attribute), 552
method), 690 tooltip (AFL.automation.prepare.SampleSeriesWidget.Checkbox
to_dict() (AFL.automation.prepare.Solution method), attribute), 560
377 tooltip (AFL.automation.prepare.SampleSeriesWidget.FloatText
to_stock_objects() (AFL.automation.prepare.StockBuilderWidgetdStoithBuilderWidget_Model
method), 67, 627, 628 tooltip (AFL.automation.prepare.SampleSeriesWidget.HBox
toggleChannels() (AFL.automation.loading.MultiChannelRelay.MultiChuta),eBR8lay
method), 37, 251 tooltip (AFL.automation.prepare.SampleSeriesWidget.IntText
toggleChannels()(AFL.automation.loading.SainSmartRelay.MultiGttaibudeRetas)
method), 318 tooltip (AFL.automation.prepare.SampleSeriesWidget.Label
toggleChannels()(AFL.automation.loading.SainSmartRelay.SainSmartRelay.594
method), 47, 319, 320 tooltip (AFL.automation.prepare.SampleSeriesWidget.Text
toJSON() (AFL.automation.APIServer.Driver.PersistentConfig attribute), 614
method), 161 tooltip (AFL.automation.prepare.SampleSeriesWidget.VBox
toJSON() (AFL.automation.shared.PersistentConfig.PersistentConfig attribute), 622
method), 77, 746, 748 tooltip (AFL.automation.prepare.SweepBuilderWidget.Button
token_in_blocklist_loader() attribute), 635
(AFL.automation.APIServer.APIServer.JWTManagooltip (AFL.automation.prepare.SweepBuilderWidget.Checkbox
method), 131 attribute), 643
token_verification_failed_loader() tooltip(AFL.automation.prepare.SweepBuilderWidget.HBox
(AFL.automation.APIServer.APIServer.JWTManager attribute), 650
method), 131 tooltip (AFL.automation.prepare.SweepBuilderWidget.Label
token_verification_loader() attribute), 658
(AFL.automation.APIServer.APIServer.JWTManagooltip (AFL.automation.prepare.SweepBuilderWidget.Text
method), 131 attribute), 678
tooltip (AFL.automation.prepare.DeckBuilderWidget.Buttbooltip (AFL.automation.prepare.SweepBuilderWidget.VBox
attribute), 391 attribute), 685
tooltip (AFL.automation.prepare.DeckBuilderWidget.Che tklp (AFL.automation.prepare.DeckBuilderWidget.Layout
attribute), 399 attribute), 433
tooltip (AFL.automation.prepare.DeckBuilderWidget.Dropdow(AFL.automation.prepare.PrepareWidget.Layout at-
attribute), 412 tribute), 518
tooltip (AFL.automation.prepare.DeckBuilderWidget.HBc\top (AFL.automation.prepare.SampleSeriesWidget.Layout
attribute), 420 attribute), 599
tooltip (AFL.automation.prepare.DeckBuilderWidget.Lababp (AFL.automation.prepare.SweepBuilderWidget.Layout
attribute), 428 attribute), 663
tooltip (AFL.automation.prepare.DeckBuilderWidget.Texttouch() (AFL.automation.instrument.SeabreezeUVVis.Path
attribute), 445 method), 200
tooltip(AFL.automation.prepare.DeckBuilderWidget.VBoxprint() (in module AFL.automation.shared.utilities),
attribute), 453 81, 784
tooltip(AFL.automation.prepare.PrepareWidget.Button trait_defaults()(AFL.automation.prepare.DeckBuilderWidget.Button
attribute), 480 method), 391
tooltip (AFL.automation.prepare.PrepareWidget.Checkbotrait_defaults() (AFL.automation.prepare.DeckBuilderWidget.Checkbotrait_defaults() (AFL.automation.prepare.DeckBuilderWidget.Checkbotrait_default
attribute), 488 method), 399
tooltip (AFL.automation.prepare.PrepareWidget.Dropdowtrait_defaults() (AFL.automation.prepare.DeckBuilderWidget.Dropdo
attribute), 497 method), 412
tooltip (AFL.automation.prepare.PrepareWidget.HBox trait_defaults() (AFL.automation.prepare.DeckBuilderWidget.HBox

 $\verb|tooltip| (AFL. automation. prepare. Prepare Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. Label | \verb|trait_defaults()| (AFL. automation. prepare. Deck Builder Widget. De$

method), 420

method), 428

attribute), 505

attribute), 513

- trait_defaults() (AFL.automation.prepare.DeckBuilderWidget_Eugants() (AFL.automation.prepare.DeckBuilderWidget.Button method), 437 class method), 391
- trait_defaults() (AFL.automation.prepare.DeckBuilderWidget.Ewents() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 446 class method), 399
- trait_defaults() (AFL.automation.prepare.DeckBuilderWidget.EvBents() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 453 class method), 413
- trait_defaults() (AFL.automation.prepare.PrepareWidgataHuttervents() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 480 class method), 420
- trait_defaults() (AFL.automation.prepare.PrepareWidgatailte.debents() (AFL.automation.prepare.DeckBuilderWidget.Label method), 488 class method), 428
- trait_defaults() (AFL.automation.prepare.PrepareWidgatalitopekents() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 497 class method), 437
- trait_defaults() (AFL.automation.prepare.PrepareWidgatalBoevents() (AFL.automation.prepare.DeckBuilderWidget.Text method), 505 class method), 446
- trait_defaults() (AFL.automation.prepare.PrepareWidgatdixtbetvents() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 513 class method), 453
- trait_defaults() (AFL.automation.prepare.PrepareWidgetAidtyoewents() (AFL.automation.prepare.PrepareWidget.Button method), 522 class method), 480
- trait_defaults() (AFL.automation.prepare.PrepareWidgetaFext_events() (AFL.automation.prepare.PrepareWidget.Checkbox method), 536 class method), 488
- trait_defaults() (AFL.automation.prepare.PrepareWidgataMRozevents() (AFL.automation.prepare.PrepareWidget.Dropdown method), 543 class method), 498
- trait_defaults() (AFL.automation.prepare.SampleSeriet WädgeteWeintus() (AFL.automation.prepare.PrepareWidget.HBox method), 552 class method), 505
- trait_defaults() (AFL.automation.prepare.SampleSerietWailgeteGhecksOx (AFL.automation.prepare.PrepareWidget.Label method), 560 class method), 513
- trait_defaults() (AFL.automation.prepare.SampleSeriex WailgeteWentExx) (AFL.automation.prepare.PrepareWidget.Layout method), 571 class method), 522
- trait_defaults() (AFL.automation.prepare.SampleSerietWaidgeteWBnts() (AFL.automation.prepare.PrepareWidget.Text method), 578 class method), 536
- trait_defaults() (AFL.automation.prepare.SampleSerietWailgeteWneFets() (AFL.automation.prepare.PrepareWidget.VBox method), 586 class method), 543
- method), 586 class method), 543 trait_defaults() (AFL.automation.prepare.SampleSeriesWidget.Button
- method), 594 class method), 552
 trait_defaults() (AFL.automation.prepare.SampleSeries Wait yet Novus () (AFL.automation.prepare.SampleSeries Widget.Checkbox method), 604 class method), 560
- trait_defaults() (AFL.automation.prepare.SampleSeriexWadgetEleants() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 615 class method), 571
- trait_defaults() (AFL.automation.prepare.SampleSerie*Waits() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 622 class method), 578
- trait_defaults() (AFL.automation.prepare.SweepBuilderWällgerBattos() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 635 class method), 587
- trait_defaults() (AFL.automation.prepare.SweepBuilder\didgeeCentklox(AFL.automation.prepare.SampleSeriesWidget.Label method), 643 class method), 594
- trait_defaults() (AFL.automation.prepare.SweepBuilder Wällgere Method), 650 class method), 604
- trait_defaults() (AFL.automation.prepare.SweepBuilder Wailger Lebtds() (AFL.automation.prepare.SampleSeriesWidget.Text method), 658 class method), 615
- trait_defaults() (AFL.automation.prepare.SweepBuilder Weitgeelkaytus() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 668 class method), 622
- trait_defaults() (AFL.automation.prepare.SweepBuilderWidget.Button method), 678 class method), 635
- trait_defaults() (AFL.automation.prepare.SweepBuilderWädgerVBnts() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 685 class method), 643

- trait_events() (AFL.automation.prepare.SweepBuilderWindgistHBas_value() (AFL.automation.prepare.SampleSeriesWidget.Layou class method), 650 method), 604
- trait_events() (AFL.automation.prepare.SweepBuilderWidgitLhbs_value() (AFL.automation.prepare.SampleSeriesWidget.Text class method), 658 method), 615
- trait_events() (AFL.automation.prepare.SweepBuilderWindgitLhasutvalue() (AFL.automation.prepare.SampleSeriesWidget.VBox class method), 668 method), 622
- trait_events() (AFL.automation.prepare.SweepBuilderWidgettThats_value() (AFL.automation.prepare.SweepBuilderWidget.Buttoclass method), 678

 method), 635
- trait_events() (AFL.automation.prepare.SweepBuilderWidget.VBas_value() (AFL.automation.prepare.SweepBuilderWidget.Chec class method), 685 method), 643
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgehBsttvalue() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 392 method), 651
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgeh&evkBoue() (AFL.automation.prepare.SweepBuilderWidget.Labe method), 400 method), 658
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgehBsopddwe() (AFL.automation.prepare.SweepBuilderWidget.Layo method), 413 method), 668
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidget.Text method), 420 method), 420 method), 678
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgethBbbblalue() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 428 method), 685
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidget.Button method), 437 method), 392
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgetEtxatdata() (AFL.automation.prepare.DeckBuilderWidget.Checkbuilderwidget.Checkbuilderwi
- trait_has_value() (AFL.automation.prepare.DeckBuilderWidgem&Bodata() (AFL.automation.prepare.DeckBuilderWidget.Dropdo method), 453 method), 413
- trait_has_value() (AFL.automation.prepare.PrepareWidgatiButmertadata() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 481 method), 420
- trait_has_value() (AFL.automation.prepare.PrepareWidgatiChandchadata() (AFL.automation.prepare.DeckBuilderWidget.Label method), 489 method), 428
- trait_has_value() (AFL.automation.prepare.PrepareWidgatiPropakadata() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 498 method), 438
- trait_has_value() (AFL.automation.prepare.PrepareWidgatiHBmetadata() (AFL.automation.prepare.DeckBuilderWidget.Text method), 505 method), 446
- trait_has_value() (AFL.automation.prepare.PrepareWidgetiltalmetadata() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 513 method), 454
- trait_has_value()(AFL.automation.prepare.PrepareWidgetiltaymertadata()(AFL.automation.prepare.PrepareWidget.Button method), 522 method), 481
- trait_has_value() (AFL.automation.prepare.PrepareWidgetiffexmetadata() (AFL.automation.prepare.PrepareWidget.Checkbox method), 536 method), 489
- trait_has_value() (AFL.automation.prepare.PrepareWidgativBmetadata() (AFL.automation.prepare.PrepareWidget.Dropdown method), 543 method), 498
- trait_has_value() (AFL.automation.prepare.SampleSerierWidgmeRaddata() (AFL.automation.prepare.PrepareWidget.HBox method), 553 method), 505
- trait_has_value() (AFL.automation.prepare.SampleSerierWidgetEbekktok) (AFL.automation.prepare.PrepareWidget.Label method), 561 method), 513
- trait_has_value() (AFL.automation.prepare.SampleSerierWidgmeFladaTex() (AFL.automation.prepare.PrepareWidget.Layout method), 571 method), 523
- trait_has_value() (AFL.automation.prepare.SampleSerierWidgettBdata() (AFL.automation.prepare.PrepareWidget.Text method), 579 method), 536
- trait_has_value() (AFL.automation.prepare.SampleSerierWidgmeHuallexta() (AFL.automation.prepare.PrepareWidget.VBox method), 587 method), 544
- trait_has_value() (AFL.automation.prepare.SampleSeriesWidgetAbdata() (AFL.automation.prepare.SampleSeriesWidget.Button method), 594 method), 553

- trait_metadata() (AFL.automation.prepare.SampleSerietWaitger(GhestD):(AFL.automation.prepare.PrepareWidget.Label method), 561 method), 513
- trait_metadata() (AFL.automation.prepare.SampleSerietWaidgetWaidgetWalkerVAFL.automation.prepare.PrepareWidget.Layout method), 571 method), 523
- trait_metadata() (AFL.automation.prepare.SampleSerietWaidgetWaidgetMares() (AFL.automation.prepare.PrepareWidget.Text method), 579 method), 536
- trait_metadata() (AFL.automation.prepare.SampleSeriexWaidgetWaidgetWantEsx() (AFL.automation.prepare.PrepareWidget.VBox method), 587 method), 544
- trait_metadata() (AFL.automation.prepare.SampleSerietWaidgetWaidgetManders() (AFL.automation.prepare.SampleSeriesWidget.Button method), 595 method), 553
- trait_metadata() (AFL.automation.prepare.SampleSeriexWaidgetNamesa() (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 604 method), 561
- trait_metadata() (AFL.automation.prepare.SampleSerie*Wädterfemes() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 615 method), 571
- trait_metadata() (AFL.automation.prepare.SampleSeriexWailgetMass() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 623 method), 579
- trait_metadata() (AFL.automation.prepare.SweepBuilder\direction{\displaystar} trait_method), 635 (AFL.automation.prepare.SampleSeriesWidget.IntText method), 587
- trait_metadata() (AFL.automation.prepare.SweepBuilder Wailgen 6thes/DotAFL.automation.prepare.SampleSeriesWidget.Label method), 643 method), 595
- trait_metadata() (AFL.automation.prepare.SweepBuilder\didgen\didg
- trait_metadata() (AFL.automation.prepare.SweepBuilder\didgenAmbel() (AFL.automation.prepare.SampleSeriesWidget.Text method), 659 method), 615
- trait_metadata() (AFL.automation.prepare.SweepBuilder Wildgenhmess(t) (AFL.automation.prepare.SampleSeriesWidget.VBox method), 668 method), 623
- trait_metadata() (AFL.automation.prepare.SweepBuilderWidget.Button method), 678 method), 635
- trait_metadata() (AFL.automation.prepare.SweepBuilderWädgen\abelaes() (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 686

 method), 643
- trait_names() (AFL.automation.prepare.DeckBuilderWidgeaButmames() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 392 method), 651
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenChenkhnes() (AFL.automation.prepare.SweepBuilderWidget.Label method), 400 method), 659
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenDropdmen() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 413 method), 668
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenHB_mames() (AFL.automation.prepare.SweepBuilderWidget.Text method), 421 method), 678
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenLubrames() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 428 method), 686
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenLtymalues() (AFL.automation.prepare.DeckBuilderWidget.Button method), 438 method), 392
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenTextvalues() (AFL.automation.prepare.DeckBuilderWidget.Checkbox method), 446 method), 400
- trait_names() (AFL.automation.prepare.DeckBuilderWidgenYBaxalues() (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 454 method), 413
- trait_names() (AFL.automation.prepare.PrepareWidget.Butteint_values() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 481 method), 421
- trait_names() (AFL.automation.prepare.PrepareWidget.Chrackbowalues() (AFL.automation.prepare.DeckBuilderWidget.Label method), 489 method), 428
- trait_names() (AFL.automation.prepare.PrepareWidget.Parapidownalues() (AFL.automation.prepare.DeckBuilderWidget.Layout method), 498 method), 438
- trait_names() (AFL.automation.prepare.PrepareWidget.HBait_values() (AFL.automation.prepare.DeckBuilderWidget.Text method), 506 method), 446

- trait_values() (AFL.automation.prepare.DeckBuilderWidgai\VBQ): (AFL.automation.prepare.DeckBuilderWidget.Dropdown method), 454 method), 413
- trait_values() (AFL.automation.prepare.PrepareWidgettHadious() (AFL.automation.prepare.DeckBuilderWidget.HBox method), 481 method), 421
- trait_values() (AFL.automation.prepare.PrepareWidgett@heicks()) (AFL.automation.prepare.DeckBuilderWidget.Label method), 489 method), 429
- trait_values() (AFL.automation.prepare.PrepareWidgett\Paipts(\mathbb{\text{Paipts}}(\mathbb{\text{P}}(AFL.automation.prepare.DeckBuilderWidget.Layout method), 498 method), 438
- trait_values() (AFL.automation.prepare.PrepareWidget#Rixs() (AFL.automation.prepare.DeckBuilderWidget.Text method), 506 method), 447
- trait_values() (AFL.automation.prepare.PrepareWidgetMrdiets() (AFL.automation.prepare.DeckBuilderWidget.VBox method), 513 method), 454
- trait_values() (AFL.automation.prepare.PrepareWidget:Hungirus() (AFL.automation.prepare.PrepareWidget.Button method), 523 method), 481
- trait_values() (AFL.automation.prepare.PrepareWidgetfFeaits() (AFL.automation.prepare.PrepareWidget.Checkbox method), 536 method), 489
- trait_values() (AFL.automation.prepare.PrepareWidgett\dats() (AFL.automation.prepare.PrepareWidget.Dropdown method), 544 method), 498
- trait_values() (AFL.automation.prepare.SampleSeriesWidgettBut) (AFL.automation.prepare.PrepareWidget.HBox method), 553 method), 506
- trait_values() (AFL.automation.prepare.SampleSeriesWidgettSk\dAbE.automation.prepare.PrepareWidget.Label method), 561 method), 514
- trait_values() (AFL.automation.prepare.SampleSeriesWidgettElQu(AEL.automation.prepare.PrepareWidget.Layout method), 571 method), 523
- trait_values() (AFL.automation.prepare.SampleSeriesWidgettIsR)x(AFL.automation.prepare.PrepareWidget.Text method), 579 method), 537
- trait_values() (AFL.automation.prepare.SampleSeriesWidgattls(Te(AFL.automation.prepare.PrepareWidget.VBox method), 587 method), 544
- trait_values() (AFL.automation.prepare.SampleSeriesWidget.Button method), 595 method), 553
- trait_values() (AFL.automation.prepare.SampleSeriesWidgettExty) (AFL.automation.prepare.SampleSeriesWidget.Checkbox method), 604 method), 561
- trait_values() (AFL.automation.prepare.SampleSeriesWidget.FloatText method), 615 method), 572
- trait_values() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 623 method), 579
- trait_values() (AFL.automation.prepare.SweepBuilderWidgettBU)t6AFL.automation.prepare.SampleSeriesWidget.IntText method), 636 method), 587
- trait_values() (AFL.automation.prepare.SweepBuilderWidgettElgetMcMc.automation.prepare.SampleSeriesWidget.Label method), 643 method), 595
- trait_values() (AFL.automation.prepare.SweepBuilderWirkzitHBokAFL.automation.prepare.SampleSeriesWidget.Layout method), 651 method), 605
- trait_values() (AFL.automation.prepare.SweepBuilderWidgettEaDe(AFL.automation.prepare.SampleSeriesWidget.Text method), 659 method), 616
- trait_values() (AFL.automation.prepare.SweepBuilderWirdgittb@)dutFL.automation.prepare.SampleSeriesWidget.VBox method), 668 method), 623
- trait_values() (AFL.automation.prepare.SweepBuilderWinkgitEQ) (AFL.automation.prepare.SweepBuilderWidget.Button method), 678 method), 636
- trait_values() (AFL.automation.prepare.SweepBuilderWidget.VB) (AFL.automation.prepare.SweepBuilderWidget.Checkbox method), 686 method), 644
- traits() (AFL.automation.prepare.DeckBuilderWidget.Buttomatis() (AFL.automation.prepare.SweepBuilderWidget.HBox method), 392 method), 651
- traits() (AFL.automation.prepare.DeckBuilderWidget.Cherkbicas() (AFL.automation.prepare.SweepBuilderWidget.Label method), 400 method), 659

```
traits()(AFL.automation.prepare.SweepBuilderWidget.Layout
                                                                                                                                                                                                                                                                                                                                   54, 353, 360
                                              method), 669
                                                                                                                                                                                                                                                                                     type (AFL.automation.APIServer.APIServer.ServiceInfo
traits() (AFL.automation.prepare.SweepBuilderWidget.Text
                                                                                                                                                                                                                                                                                                                                   attribute), 141
                                              method), 679
                                                                                                                                                                                                                                                                                     {\tt type}\, (AFL. automation. shared. Server Discovery. Async Service Info
traits()(AFL.automation.prepare.SweepBuilderWidget.VBox
                                                                                                                                                                                                                                                                                                                                   attribute), 753
                                             method), 686
                                                                                                                                                                                                                                                                                     type (AFL.automation.shared.ServerDiscovery.ServiceInfo
transfer() (AFL.automation.EpicsADLiveProcess.Client.Client
                                                                                                                                                                                                                                                                                                                                   attribute), 770
                                              method), 24, 792, 793
                                                                                                                                                                                                                                                                                     types (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
transfer() (AFL.automation.loading.OneSelectorBlowoutSampleCellt@haselect@rBlowoutSampleCell
                                                                                                                                                                                                                                                                                    types (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                                              method), 263
transfer() (AFL.automation.loading.OneSelectorBlowoutSampleCedtfFiboSe)e766rBlowoutSampleCell
                                              method), 266
transfer() (AFL.automation.loading.PushPullSelectorSampleCell.PushPullSelectorSampleCell
                                             method), 44, 299, 304
                                                                                                                                                                                                                                                                                    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.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.RsoXSolutionSam
                                              method), 46, 310, 316
                                                                                                                                                                                                                                                                                                                                   56, 362, 364
transfer() (AFL.automation.loading.TwoSelectorBlowoutSampleCell TwpSelectorBlowoutSampleCell
                                             method), 55, 356, 361
                                                                                                                                                                                                                                                                                                                                   (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), 62, 460, 462
                                                                                                                                                                                                                                                                                    unlatch_shaker()(AFL.automation.prepare.Dummy_OT2_Driver.Dumm
transfer() (AFL.automation.prepare.OT2Client.OT2Client
                                                                                                                                                                                                                                                                                                                                   method), 61, 460, 462
                                             method), 62, 467, 470
                                                                                                                                                                                                                                                                                     unlink() (AFL.automation.instrument.SeabreezeUVVis.Path
transfer() (AFL.automation.sample.CastingServer.OT2Client
                                                                                                                                                                                                                                                                                                                                   method), 201
                                              method), 703
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.DeckBuilderWidget.Button
transform() (AFL.automation.shared.DataLabelerWidget.OrdinalEngodera), 392
                                             method), 719
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.DeckBuilderWidget.Checkbox
transform() (AFL. automation. shared. Diffraction Labeler. Ordinal Engage transform(), 400
                                             method), 739
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.DeckBuilderWidget.Dropdown
transmit() (AFL.automation.APIServer.QueueDaemon.DataTrashcanethod), 414
                                             method), 178
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.DeckBuilderWidget.HBox
trap_http_exception()
                                                                                                                                                                                                                                                                                                                                   method), 421
                                              (AFL.automation.APIServer.APIServer.Flask
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.DeckBuilderWidget.Label
                                             method), 115
                                                                                                                                                                                                                                                                                                                                   method), 429
tubing (AFL. automation. loading. PushPull Selector Sample \textit{GalbBuking} () (AFL. automation. prepare. Deck Builder Widget. Layout tubing (AFL. automation. prepare. Deck Builder Widget. Dec
                                              attribute), 301
                                                                                                                                                                                                                                                                                                                                   method), 438
tubing (AFL. automation. loading. RSoXSS olution Sample Cell Tobing (AFL. automation. prepare. Deck Builder Widget. Text to the same prepare of 
                                             attribute), 313
                                                                                                                                                                                                                                                                                                                                   method), 447
tubing
                                               (AFL.automation.loading.Tubing.Tubing
                                                                                                                                                                                                                                                                                   unobserve() (AFL.automation.prepare.DeckBuilderWidget.VBox
                                              tribute), 54, 349
                                                                                                                                                                                                                                                                                                                                   method), 454
tubing (AFL. automation. loading. Two Selector Blowout Sample Gall Twicks) (AFL. automation. prepare Widget. Button the first of the 
                                              attribute), 353
                                                                                                                                                                                                                                                                                                                                   method), 482
\textbf{Tubing} (class \ in AFL. automation. loading. PushPull Selector \textbf{SampleCeVO}) (AFL. automation. prepare. Prepare Widget. Checkbox and the property of the
                                                                                                                                                                                                                                                                                                                                   method), 489
\textbf{Tubing} \ (class\ in\ AFL. automation. loading. RSoXSSolution Sample Sell ve ()\ (AFL. automation. prepare. Prepare Widget. Dropdown and the sell very s
                                                                                                                                                                                                                                                                                                                                   method), 499
Tubing (class in AFL.automation.loading.Tubing), 54, unobserve() (AFL.automation.prepare.PrepareWidget.HBox
                                                                                                                                                                                                                                                                                                                                   method), 506
\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
                                              353
                                                                                                                                                                                                                                                                                                                                   method), 514
TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                      (class
                                                                                                                                                                                                                                                                in unobserve() (AFL.automation.prepare.PrepareWidget.Layout
                                             AFL.automation.loading.OneSelectorBlowoutSampleCell), method), 523
                                                                                                                                                                                                                                                                                     unobserve() (AFL.automation.prepare.PrepareWidget.Text
TwoSelectorBlowoutSampleCell
                                                                                                                                                                                                       (class
                                                                                                                                                                                                                                                                 in
                                                                                                                                                                                                                                                                                                                                   method), 537
                                             AFL.automation.loading.TwoSelectorBlowoutSampleCell).
```

```
unobserve() (AFL.automation.prepare.PrepareWidget.VBunobserve_all() (AFL.automation.prepare.PrepareWidget.Dropdown method), 544 method), 499
unobserve() (AFL.automation.prepare.SampleSeriesWidgetnBiaserve_all() (AFL.automation.prepare.PrepareWidget.HBox method), 553 method), 506
```

- unobserve() (AFL.automation.prepare.SampleSeriesWidgetnCheckboer_all() (AFL.automation.prepare.PrepareWidget.Label method), 561 method), 514
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetnblosseFixee_all() (AFL.automation.prepare.PrepareWidget.Layout method), 572
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetndBserve_all() (AFL.automation.prepare.PrepareWidget.Text method), 580 method), 537
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetrNobEerrve_all() (AFL.automation.prepare.PrepareWidget.VBox method), 588 method), 545
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetNoblsedrve_all() (AFL.automation.prepare.SampleSeriesWidget.Button method), 595

 method), 595
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetrlchpseutve_all() (AFL.automation.prepare.SampleSeriesWidget.Checkbomethod), 605

 method), 562
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetrEbserve_all() (AFL.automation.prepare.SampleSeriesWidget.FloatTe. method), 616 method), 572
- unobserve() (AFL.automation.prepare.SampleSeriesWidgetnWBaerve_all() (AFL.automation.prepare.SampleSeriesWidget.HBox method), 580
- unobserve() (AFL.automation.prepare.SweepBuilderWidgenBbserve_all() (AFL.automation.prepare.SampleSeriesWidget.IntText method), 636 method), 588
- unobserve() (AFL.automation.prepare.SweepBuilderWidgan6hsekbrec_all() (AFL.automation.prepare.SampleSeriesWidget.Label method), 644 method), 596
- unobserve() (AFL.automation.prepare.SweepBuilderWidganblbsarve_all() (AFL.automation.prepare.SampleSeriesWidget.Layout method), 651

 method), 605

 unobserve() (AFL automation prepare SweepBuilderWidgenblbbebryo all() (AFL automation prepare SampleSeriesWidget Text
- unobserve() (AFL.automation.prepare.SweepBuilderWidgenbbserve_all() (AFL.automation.prepare.SampleSeriesWidget.Text method), 659 method), 616
- unobserve() (AFL.automation.prepare.SweepBuilderWidganbbsænve_all() (AFL.automation.prepare.SampleSeriesWidget.VBox method), 669 method), 624
- unobserve() (AFL.automation.prepare.SweepBuilderWidgenBoserve_all() (AFL.automation.prepare.SweepBuilderWidget.Button method), 679 method), 636
- unobserve() (AFL.automation.prepare.SweepBuilderWidgenVBserve_all() (AFL.automation.prepare.SweepBuilderWidget.Checkbe method), 686 method), 644 unobserve_all() (AFL.automation.prepare.DeckBuilderWingetsButtenall() (AFL.automation.prepare.SweepBuilderWidget.HBox
- method), 393 method), 652
- unobserve_all() (AFL.automation.prepare.DeckBuilderWindgbs&hvekladxl() (AFL.automation.prepare.SweepBuilderWidget.Label method), 401 method), 660
- unobserve_all() (AFL.automation.prepare.DeckBuilderWindgetsDrvpdaWh() (AFL.automation.prepare.SweepBuilderWidget.Layout method), 414 method), 669
- unobserve_all() (AFL.automation.prepare.DeckBuilderWinkgesEhBrex_all() (AFL.automation.prepare.SweepBuilderWidget.Text method), 421 method), 679
- unobserve_all() (AFL.automation.prepare.DeckBuilderWindgbsEarbel_all() (AFL.automation.prepare.SweepBuilderWidget.VBox method), 429 method), 687
- unobserve_all() (AFL.automation.prepare.DeckBuilderWindgicklinggError, 781
 - method), 439 unqueued() (AFL.automation.APIServer.Driver.Driver
- unobserve_all() (AFL.automation.prepare.DeckBuilderWidget.Textmethod), 19, 158, 162
 - method), 447 unqueued() (AFL.automation.APIServer.DummyDriver.Driver
- unobserve_all() (AFL.automation.prepare.DeckBuilderWidget.VBonethod), 164
 - method), 455 unqueued() (AFL.automation.APIServer.DummyDriver.DummyDriver
- ${\tt unobserve_all()} \ (AFL. automation. prepare. Prepare Widget. Button \ method), \ 169$
 - method), 482 unqueued() (AFL.automation.APIServer.DummyOT2Driver.Driver
- unobserve_all() (AFL.automation.prepare.PrepareWidget.Checkbonethod), 171
- method), 490 unqueued() (AFL.automation.APIServer.DummyOT2Driver.DummyDriver

method), 175	method), 700
unqueued() (AFL.automation.instrument.DummySAS.Drivænqueue	
method), 184	method), 706
unqueued() (AFL.automation.instrument.DummySAS.Dumunquese	d() (AFL.automation.sample_env.TemperatureDeck.Temperature
method), 188	method), 710
unqueued() (AFL.automation.instrument.I22SAXS.Driver unqueue	d_base() (AFL.automation.APIServer.Client.Client
method), 189	method), 18, 153, 155
unqueued() (AFL.automation.instrument.I22SAXS.I22SAXSinqueue	
method), 193	method), 234
unqueued() (AFL.automation.instrument.SeabreezeUVVis.Darqueeue	d_base() (AFL.automation.prepare.DeckBuilderWidget.Client
method), 195	method), 403
unqueued() (AFL.automation.instrument.SeabreezeUVVis.Singlweese	eUVXse() (AFL.automation.prepare.OT2Client.Client
method), 206	method), 465
unqueued() (AFL.automation.instrument.SpecScreen_DrivenDurbue	d_base() (AFL.automation.prepare.OT2Client.OT2Client
method), 209	method), 469
unqueued() (AFL.automation.instrument.SpecScreen_DrivenSpecSc	debas@fi)&AFL.automation.prepare.SampleSeriesWidget.Client
method), 212	method), 564
unqueued() (AFL.automation.loading.LoadStopperDriver.Dniqueue	d_base() (AFL.automation.sample.CastingServer.Client
method), 236	method), 698
unqueued() (AFL.automation.loading.LoadStopperDriver.Lunquberage	plehlasive(t) (AFL.automation.sample.CastingServer.OT2Client
method), 241	method), 705
unqueued() (AFL.automation.loading.OneSelectorBlowoutSmrqdie	โซฟิป_สโซ้r_services()
method), 257	(AFL.automation.APIServer.APIServer.Zeroconf
unqueued() (AFL.automation.loading.OneSelectorBlowoutSampleC	CallethoaSel&AorBlowoutSampleCell
method), 263 unregis	ter_all_services()
unqueued() (AFL.automation.loading.OneSelectorBlowoutSampleC	EdlA FlyaStelenototiBhoshourtSlaSepheGEllscovery.Zeroconf
method), 267	method), 778
unqueued() (AFL.automation.loading.PneumaticPressureSumpelgCs	ttenriservice()
method), 271	(AFL.automation.APIServer.APIServer.Zeroconf
$unqueued () \ (AFL. automation. loading. Pneumatic Pressure Sample Central Control of $	elh ılethma h)atit&PressureSampleCell
	ter_service()
$unqueued () \ (AFL. automation. loading. Pneumatic Sample Cell. Driven and the property of t$	
method), 281	method), 777
unqueued() (AFL.automation.loading.PneumaticSampleCathBaturi	*
method), 286	AFL.automation.APIServer.APIServer), 89
unqueued() (AFL.automation.loading.PushPullSelectorSamplaCtell	DrAEL.automation.APIServer.Driver.PersistentConfig
method), 295	method), 161
$unqueued () \ (AFL. automation. loading. PushPull Selector Sample Gell () \ (AFL. automation. loading. PushPull Selector Sample Gell () \ () \ () \ () \ () \ () \ () \ ()$	PuAlFRuduStehnattivStaAnpl&Eveler.QueueDaemon.DataTrashcan
method), 301	method), 179
unqueued() (AFL.automation.loading.RSoXSSolutionSamplipCelt.D	\red{p} (ear FL. automation. loading. One Selector Blowout Sample Cell. defa
method), 306	method), 269
unqueued() (AFL.automation.loading.RSoXSSolutionSamplipCeltisC	
method), 312	method), 279
unqueued() (AFL.automation.loading.TwoSelectorBlowoutSpringle(
method), 351	method), 288
unqueued() (AFL.automation.loading.TwoSelectorBlowoutSpringle(
method), 358	method), 303
unqueued() (AFL.automation.prepare.Dummy_OT2_DriveupDate()	· · · · · · · · · · · · · · · · · · ·
method), 456	method), 315
unqueued() (AFL.automation.prepare.Dummy_OT2_Driveupdiane)	
method), 461	method), 360
$unqueued () \ (AFL. automation. sample. Casting Server. Casting {\it Scharter} () \ (AFL. automation. sample. Casting {\it Scharter} () \ () \ () \ () \ () \ () \ () \ () $	2 (1
<pre>method), 696 unqueued() (AFL.automation.sample.CastingServer.Driverupdate()</pre>	method), 730

```
method), 744
                                                                                                                                                                                                                              method), 73, 725, 731
update() (AFL.automation.shared.PersistentConfig.Persistent@protocol_order_preview_cb()
                               method), 77, 747, 748
                                                                                                                                                                                                                               (AFL.automation.prepare.PrepareWidget.SampleSeriesWidget
update_colors() (AFL.automation.shared.DatasetWidget.DatasetWidgetbod), 526
                               method), 73, 725, 731
                                                                                                                                                                                              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), 65, 606, 624
                               method), 74, 727, 732
                                                                                                                                                                                              update_record() (AFL.automation.APIServer.APIServer.ServiceInfo
update_component_row_cb()
                                                                                                                                                                                                                               method), 144
                               (AFL.automation.prepare.PrepareWidget.SweepBuibdarWeidgetord() (AFL.automation.shared.ServerDiscovery.AsyncService
                               method), 529
                                                                                                                                                                                                                              method), 751
update_component_row_cb()
                                                                                                                                                                                              update_record() (AFL.automation.shared.ServerDiscovery.AsyncService
                               (AFL.automation.prepare.SweepBuilderWidget.SweepBuildenWildget), 756
                               method), 68, 670, 687
                                                                                                                                                                                              update_record() (AFL.automation.shared.ServerDiscovery.ServiceBrown
update_composition_colors()
                                                                                                                                                                                                                               method), 768
                               (AFL.automation.shared.DataLabelerWidget.DataIpdarlerWieword() (AFL.automation.shared.ServerDiscovery.ServiceInfo
                               method), 71, 713, 722
                                                                                                                                                                                                                              method), 773
update_composition_plot()
                                                                                                                                                                                              update_sample_dim()
                               (AFL. automation. shared. Dataset Widget. Dataset Widget
                                                                                                                                                                                                                              (AFL.automation.shared.DatasetWidget.DatasetWidget
                               method), 73, 725, 731
                                                                                                                                                                                                                              method), 73, 726, 731
update_deck_graphic_cb()
                                                                                                                                                                                              update_scattering_plot()
                               (AFL.automation.prepare.DeckBuilderWidget.DeckBuilderWAfgetautomation.shared.DatasetWidget.DatasetWidget
                                                                                                                                                                                                                               method), 73, 725, 731
                               method), 60, 404, 455
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), 74, 728, 732
                               method), 491
                                                                                                                                                                                              update_service() (AFL.automation.APIServer.APIServer.Zeroconf
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget147
                                                                                                                                                                                              \verb"update_service" () (AFL. automation. shared. Server Discovery. Async Zerocomunication and the property of 
                               method), 73, 725, 731
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget_Tview
                                                                                                                                                                                              \verb"update_service" () (AFL. automation. shared. Server Discovery. Zeroconf
                               method), 74, 728, 732
update_extract_coords()
                                                                                                                                                                                                                               method), 777
                               (AFL.automation.shared.DatasetWidget.DatasetWidgetate_sort_order_cb()
                                                                                                                                                                                                                               (AFL. automation. prepare. Prepare Widget. Sample Series Widget
                               method), 73, 726, 731
update_location_check_cb()
                                                                                                                                                                                                                               method), 527
                               (AFL.automation.prepare.PrepareWidget.StockBuilpdaWidgeort_order_cb()
                               method), 528
                                                                                                                                                                                                                               (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), 65, 607, 624
                               (AFL.automation.prepare.StockBuilderWidget.StockBuilder\(\text{StockBuilder}\)\(\text{StockBuilder}\)\(\text{AFL.automation.loading.LoadStopperDriver.StopLoad\)
                               method), 66, 626, 628
                                                                                                                                                                                                                               method), 247
update_mixing_well_preview_cb()
                                                                                                                                                                                              update_status() (AFL.automation.loading.LoadStopperDriver.StopLoad
                               (AFL.automation.prepare.PrepareWidget.SampleSeriesWidgetthod), 250
                               method), 527
                                                                                                                                                                                              update_status() (AFL.automation.loading.SensorCallbackThread.Sensor
update_mixing_well_preview_cb()
                                                                                                                                                                                                                               method), 50, 330, 341
                               (AFL.automation.prepare.SampleSeriesWidget.SampdeSee_iesWeitlges() (AFL.automation.loading.SensorCallbackThread.Simple
                               method), 65, 607, 624
                                                                                                                                                                                                                               method), 334
update_plot() (AFL.automation.shared.DataLabelerWidgmadateLsbakers*(@m(AFL.automation.loading.SensorCallbackThread.StopL
                               method), 71, 712, 722
                                                                                                                                                                                                                               method), 337
update_plot() (AFL.automation.shared.DataLabelerWidgendDateLsbateusWidleAFL.automation.loading.SensorCallbackThread.StopL
                               method), 71, 714, 721
                                                                                                                                                                                                                               method), 340
update_plot() (AFL.automation.shared.DiffractionLabelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffractionLubelandiffraction
                               method), 75, 733, 741
                                                                                                                                                                                                                               method), 82, 695, 705
update_plot() (AFL.automation.shared.DiffractionLabelarpdiffractionpdatalerVientext()
                               method), 75, 735, 741
                                                                                                                                                                                                                               (AFL.automation.APIServer.APIServer.Flask
update_plots() (AFL.automation.shared.DatasetWidget.DatasetWidgethod), 109
```

update_ternary_colors() (AFL.automation.shared.DiffractionLabeler.Diffi	
method), 75, 735, 742 Updated (AFL.automation.shared.ServerDiscovery.Service	<pre>validate_sweep_cb() eStateChangAFL.automation.prepare.PrepareWidget.SweepBuilderWidget</pre>
attribute), 773	method), 529
url_build_error_handlers	<pre>validate_sweep_cb()</pre>
(AFL. automation. APIS erver. APIS erver. Flask	(AFL. automation. prepare. Sweep Builder Widget. Widget. Sweep Builder Widget. Sweep Builder Widget. Sweep Builder Widget. Widget. Sweep Builder Widget.
attribute), 106	method), 67, 670, 687
url_default_functions	$value \ (\textit{AFL}. \textit{automation}. \textit{instrument}. \textit{SeabreezeUVV} is. \textit{Eq}$
(AFL. automation. API Server. API Server. Flask	attribute), 197
attribute), 124	${\tt value} (AFL. automation. prepare. Deck Builder Widget. Check box$
url_defaults() (AFL.automation.APIServer.APIServer.	
method), 122	${\tt value} (AFL. automation. prepare. Deck Builder Widget. Drop down$
url_for() (AFL.automation.APIServer.APIServer.Flask	attribute), 414
method), 117	${\tt value} (AFL. automation. prepare. Deck Builder Widget. Label$
url_map (AFL.automation.APIServer.APIServer.Flask	attribute), 429
attribute), 107	${\tt value}(AFL. automation. prepare. Deck Builder Widget. Text$
$\verb"url_map_class" (AFL. automation. APIS erver. APIS erver. Figure 1) and the properties of the prope$	lask attribute), 447
attribute), 105	${\tt value} (AFL. automation. prepare. Prepare Widget. Checkbox$
url_rule_class (AFL. automation. APIServer. APISER.	Flask attribute), 490
attribute), 105	${\tt value} (AFL. automation. prepare. Prepare Widget. Drop down$
url_value_preprocessor()	attribute), 499
(AFL.automation.APIServer.APIServer.Flask	value (AFL.automation.prepare.PrepareWidget.Label
method), 123	attribute), 514
url_value_preprocessors	value (AFL.automation.prepare.PrepareWidget.Text at-
(AFL. automation. APIS erver. APIS erver. Flask	tribute), 537
attribute), 124	${\tt value} (AFL. automation. prepare. Sample Series Widget. Checkbox$
$usbrelay {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Sain Smart Research)} and {\it (in module AFL. automation. loading. Smart Research)} and {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} and {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. Smart Research)} are {\it (in module AFL. automation. loading. loading. loading. Smart Research)} are {\it (in module AFL. automation. loading. loading$	Relay), attribute), 562
47, 316, 320	${\tt value} (AFL. automation. prepare. Sample Series Widget. Float Text$
${\tt use_x_sendfile} \ (AFL. automation. APIS erver. A$	Flask attribute), 572
property), 104	value (AFL.automation.prepare.SampleSeriesWidget.IntText
<pre>user_identity_loader()</pre>	attribute), 588
(AFL.automation.APIServer.APIServer.JWTMan	a ya lue (AFL.automation.prepare.SampleSeriesWidget.Label
method), 132	attribute), 596
<pre>user_lookup_error_loader()</pre>	value (AFL.automation.prepare.SampleSeriesWidget.Text
(AFL.automation.APIServer.APIServer.JWTMan	ager attribute), 616
method), 132	value (AFL.automation.prepare.SweepBuilderWidget.Checkbox
<pre>user_lookup_loader()</pre>	attribute), 644
(AFL.automation.APIServer.APIServer.JWTMan	agalue (AFL.automation.prepare.SweepBuilderWidget.Label
method), 132	attribute), 660
M	${\tt value} (AFL. automation. prepare. Sweep Builder Widget. Text$
V	attribute), 679
V40nly (AFL.automation.APIServer.APIServer.IPVersion attribute), 127	values() (AFL.automation.APIServer.Driver.PersistentConfig method), 162
V40nly (AFL.automation.shared.ServerDiscovery.IPVersion attribute), 760	method), 179 (AFL.automation.APIServer.QueueDaemon.DataTrashcan
V60nly (AFL.automation.APIServer.APIServer.IPVersion attribute), 127	values() (AFL.automation.loading.OneSelectorBlowoutSampleCell.defau method), 269
${\tt V60nly} (AFL. automation. shared. Server Discovery. IPV ersion of the content of the cont$	p_{OH} values () (AFL. automation. loading. Pneumatic Pressure Sample Cell. default
attribute), 760	method), 279
<pre>validate_sample_series()</pre>	values() (AFL.automation.loading.PneumaticSampleCell.defaultdict
(AFL.automation.prepare.Deck method),	method), 288
370	$values () \ (AFL. automation. loading. Push Pull Selector Sample Cell. default discovered by the property of the property of$

method), 303

values() (AFL.automation.loading.RSoXSSolutionSample \delta l.defaultdict
method), 315 (AFI automation APIServer Client Client
values() (AFL.automation.loading.TwoSelectorBlowoutSampleCell.defaultidi, 18, 153, 155
method), 360 wait() (AFL.automation.loading.LoadStopperDriver.Client
values() (AFL.automation.shared.DatasetWidget.defaultdict method), 234
method), 730 wait() (AFL.automation.prepare.DeckBuilderWidget.Client
values() (AFL.automation.shared.PersistentConfig.MutableMappingnethod), 403
method), 744 wait() (AFL automation.prepare.OT2Client.Client
values() (AFL.automation.shared.PersistentConfig.PersistentConfig method), 464 method), 747 walt() (AFL.automation.prepare.OT2Client OT2Client
Walt() (Art.automation.prepare.O12Cttent.O12Cttent
447
Wait() (AFL.automation.prepare.SampleSeriesWidget.Client method), 564
537 wait() (AFL.automation.sample.CastingServer.Client
VBox (class in AFL automation.prepare.SampleSeriesWidget), method), 698
wait() (AFL.automation.sample.CastingServer.OT2Client
VBox (class in AFL.automation.prepare.SweepBuilderWidget), method), 705
679 wait_dosage_finished()
ViciMultiposSelector (class in (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringeP
AFL.automation.loading.DoubleViciMultiposSelector), method), 32, 220, 222
227 webapp() (AFL.automation.APIServer.APIServer.APIServer
ViciMultiposSelector (class in method), 16, 92, 150
AFL.automation.loading.ViciMultiposSelector), weight (AFL.automation.APIServer.APIServer.ServiceInfo
of the state of th
view_functions (AFL.automation.APIServer.APIServer.Flaskght (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo attribute), 123
attribute), 123 visibility (AFL.automation.prepare.DeckBuilderWidget.Largett (AFL.automation.shared.ServerDiscovery.ServiceInfo
attribute), 433
visibility (AFL.automation.prepare.PrepareWidget.Layowidget_types (AFL.automation.prepare.DeckBuilderWidget.Button
attribute), 310
visibility (AFL.automation.prepare.SampleSeriesWidget, Layout_types (AFL.automation.prepare.DeckBuilderWidget.Checkbox
attribute), 000
visibility (AFL.automation.prepare.SweepBuilderWidgetwigget_types (AFL.automation.prepare.DeckBuilderWidget.Dropdown
attribute), 414
volume (AFL.automation.prepare.Component.Component widget_types (AFL.automation.prepare.DeckBuilderWidget.HBox
property), 59, 382, 384 volume (AFL.automation.prepare.factory.Solution prop-
widget_cypes (II E.automation.prepare.DeckButter Waget.Eaoct
7 (ATT
Taget_cypes (II Elamonamore perelle elamonamor
volume (AFL.automation.prepare.Solution property), attribute), 439 378 widget_types (AFL.automation.prepare.DeckBuilderWidget.Text
volume (AFL.automation.prepare.Solvent property), 380
volume() (AFL.automation.loading.PushPullSelectorSampleCell Turinges (AFL.automation.prepare.DeckBuilderWidget.VBox
melnoa), 502
volume() (AFL.automation.loading.RSoXSSolutionSample Cell Tell Tell Tell Tell Tell Tell Te
memoa), 313 attribute), 482
volume() (AFL.automation.loading.Tubing widget_types(AFL.automation.prepare.PrepareWidget.Checkbox
method), 54, 349 attribute), 490
volume() (AFL.automation.loading.TwoSelectorBlowoutSampleCell_Typies(AFL.automation.prepare.PrepareWidget.Dropdown
method), 353 volume fraction (AFL automation prepare factory Solution)
volume_fraction(AFL.automation.prepare.factory.Solution dget_types(AFL.automation.prepare.PrepareWidget.HBox property), 691 attribute), 506
and the continue (AFI) is a continue of the c
volume_fraction (AFL.automation.prepare.Solution property), 378 widget_types (AFL.automation.prepare.PrepareWidget.Label attribute) 514
000000000000000000000000000000000000000

attribute), 514

- widget_types (AFL.automation.prepare.PrepareWidget.Lowindgets (AFL.automation.prepare.PrepareWidget.Button attribute), 524 attribute), 482
- widget_types (AFL.automation.prepare.PrepareWidget.Tewidgets (AFL.automation.prepare.PrepareWidget.Checkbox attribute), 537 attribute), 490
- widget_types (AFL.automation.prepare.PrepareWidget.VBiadgets (AFL.automation.prepare.PrepareWidget.Dropdown attribute), 545 attribute), 499
- widget_types (AFL.automation.prepare.SampleSeriesWidgettsn(AFL.automation.prepare.PrepareWidget.HBox attribute), 554 attribute), 506
- widget_types (AFL.automation.prepare.SampleSeriesWidgetdGktdsk&AFL.automation.prepare.PrepareWidget.Label attribute), 562 attribute), 514
- widget_types (AFL.automation.prepare.SampleSeriesWidgetdGletdsTleArFL.automation.prepare.PrepareWidget.Layout attribute), 572 attribute), 524
- widget_types (AFL.automation.prepare.SampleSeriesWidgetdleBos (AFL.automation.prepare.PrepareWidget.Text attribute), 580 attribute), 587
- widget_types (AFL.automation.prepare.SampleSeriesWidgetClayetText(AFL.automation.prepare.PrepareWidget.VBox attribute), 588 attribute), 545
- widget_types (AFL.automation.prepare.SampleSeriesWidgettes (AFL.automation.prepare.SampleSeriesWidget.Button attribute), 596 attribute), 554
- widget_types (AFL.automation.prepare.SampleSeriesWidgetdlgestest(AFL.automation.prepare.SampleSeriesWidget.Checkbox attribute), 605 attribute), 562
- widget_types (AFL.automation.prepare.SampleSeriesWidgetSeriesWidgetSeriesWidget.FloatText attribute), 616 attribute), 572
- widget_types (AFL.automation.prepare.SampleSeriesWidgettBox (AFL.automation.prepare.SampleSeriesWidget.HBox attribute), 624 attribute), 580
- widget_types (AFL.automation.prepare.SweepBuilderWidgetdBatton(AFL.automation.prepare.SampleSeriesWidget.IntText attribute), 636 attribute), 588
- widget_types (AFL.automation.prepare.SweepBuilderWidgitlGhtsklAtL.automation.prepare.SampleSeriesWidget.Label attribute), 644 attribute), 596
- widget_types (AFL.automation.prepare.SweepBuilderWidgettBBcs:(AFL.automation.prepare.SampleSeriesWidget.Layout attribute), 652 attribute), 655
- widget_types (AFL.automation.prepare.SweepBuilderWid**girtlhehes**) (AFL.automation.prepare.SampleSeriesWidget.Text attribute), 660 attribute), 616
- widget_types (AFL.automation.prepare.SweepBuilderWidgettlbetrackAFL.automation.prepare.SampleSeriesWidget.VBox attribute), 669 attribute), 624
- widget_types (AFL.automation.prepare.SweepBuilderWidgetdGetts (AFL.automation.prepare.SweepBuilderWidget.Button attribute), 679 attribute), 636
- widget_types (AFL.automation.prepare.SweepBuilderWidgetdyBvs (AFL.automation.prepare.SweepBuilderWidget.Checkbox attribute), 687 attribute), 644
- widgets (AFL.automation.prepare.DeckBuilderWidget.Buttwidgets (AFL.automation.prepare.SweepBuilderWidget.HBox attribute), 393 attribute), 652
- widgets (AFL.automation.prepare.DeckBuilderWidget.Chewklidgets (AFL.automation.prepare.SweepBuilderWidget.Label attribute), 401 attribute), 660
- widgets (AFL.automation.prepare.DeckBuilderWidget.Drowddgets (AFL.automation.prepare.SweepBuilderWidget.Layout attribute), 414 attribute), 669
- widgets (AFL.automation.prepare.DeckBuilderWidget.HBoxidgets (AFL.automation.prepare.SweepBuilderWidget.Text attribute), 421 attribute), 679
- widgets (AFL.automation.prepare.DeckBuilderWidget.Labwidgets (AFL.automation.prepare.SweepBuilderWidget.VBox attribute), 429 attribute), 687
- widgets (AFL.automation.prepare.DeckBuilderWidget.Laywitdth (AFL.automation.prepare.DeckBuilderWidget.Layout attribute), 439 attribute), 433
- widgets (AFL.automation.prepare.DeckBuilderWidget.Textwidth (AFL.automation.prepare.PrepareWidget.Layout attribute), 447 attribute), 518
- $widgets (AFL. automation. prepare. Deck Builder Widget. VBowidth (AFL. automation. prepare. Sample Series Widget. Layout attribute), 455 \\attribute), 600$

```
width (AFL.automation.prepare.SweepBuilderWidget.Layout
        attribute), 664
with_name() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 203
with_stem() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 203
with_suffix()(AFL.automation.instrument.SeabreezeUVVis.Path
        method), 203
withdraw() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
        method), 32, 220, 222
withdraw() (AFL. automation. loading. Chemyx Syringe Pump. Syringe Pump.
        method), 221
withdraw() (AFL.automation.loading.DummyPump.DummyPump
        method), 35, 230, 231
withdraw() (AFL.automation.loading.DummyPump.SyringePump
        method), 231
withdraw() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePump
        method), 37, 253, 255
withdraw() (AFL.automation.loading.NE1kSyringePump.SyringePump
        method), 255
withdraw() (AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump
        method), 43, 292, 294
withdraw() (AFL.automation.loading.PressureControllerAsPump.SyringePump
        method), 293
withdraw() (AFL.automation.loading.SyringePump.SyringePump
        method), 54, 348
write()
              (AFL.automation.prepare.ComponentDB
        method), 368
write_bytes() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 200
write_text() (AFL.automation.instrument.SeabreezeUVVis.Path
        method), 200
wsgi_app() (AFL.automation.APIServer.APIServer.Flask
        method), 127
X
xarray_to_bytes()
                                             module
                               (in
        AFL.automation.shared.utilities), 81, 784
Ζ
zc (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
        attribute), 751
zc (AFL.automation.shared.ServerDiscovery.ServiceBrowser
        attribute), 766
Zeroconf (class in AFL.automation.APIServer.APIServer),
Zeroconf (class in AFL.automation.shared.ServerDiscovery),
         774
```