# **AFL-automation**

**NIST AFL Team** 

## **CONTENTS:**

1	Modules	3
	1.1 API Server	3
	1.2 Instrument	15
	1.3 Loading	17
	1.4 Prepare	82
	1.5 Sample	82
	1.6 Shared	82
2	API Reference	157
	2.1 AFL.automation	157
	2.2 AFL.automation.sample	
3	Module Documentation	193
	3.1 AFL.automation	193
4	Indices and tables	201
Рy	thon Module Index	203
In	dex	205

AFL-automation is a framework for instrument control. It powers the NIST AFL, but is more useful than that. It enables the easy conversion of Python classes - drivers - into robust HTTP microservices with authentication, task queueing, UI generation, data management, and more.

CONTENTS: 1

2 CONTENTS:

**CHAPTER** 

**ONE** 

#### **MODULES**

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

### 1.1 API Server

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

AFL.automation.APIServer

#### 1.1.1 AFL.automation.APIServer

#### **Modules**

Client

LoggerFilter

QueueDaemon

#### 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 ob-
                                                  ject to deposit id: str, the uuid to deposit the object
                                                  under if not specified, a new uuid will be generated
driver_status()
enqueue([interactive])
enqueued_base(**kwargs)
from_server_name(server_name, **kwargs)
get_config(name[, print_console, interactive])
get_driver_object(name)
get_object(name[, serialize])
get_queue()
get_queued_commands([inherit_commands])
get_quickbar()
get_server_time()
get_unqueued_commands([inherit_commands])
halt()
logged_in()
login(username[, populate_commands])
move_item(uuid, pos)
pause(state)
query_driver(**kwargs)
```

continues on next page

Table 4 – continued from previous page

```
queue_state()
 remove_item(uuid)
 reset_queue_daemon()
                                                  Retrieve an object from the dropbox id: str, the uuid
 retrieve_obj(uid[, delete])
                                                  of the object to retrieve delete: bool, if True, delete
                                                  the object after retrieving
 server_cmd(cmd, **kwargs)
 set_config([interactive])
 set_driver_object(**kw)
 set_object([serialize])
 unqueued_base(**kwargs)
 wait([target_uuid, interval, for_history, ...])
__init__(ip=None, port='5000', username=None, interactive=False)
classmethod from_server_name(server_name, **kwargs)
logged_in()
login(username, populate_commands=True)
driver_status()
get_queue()
wait(target_uuid=None, interval=0.1, for_history=True, first_check_delay=5.0)
get_quickbar()
server_cmd(cmd, **kwargs)
enqueued_base(**kwargs)
unqueued_base(**kwargs)
get_unqueued_commands(inherit_commands=True)
get_queued_commands(inherit_commands=True)
enqueue(interactive=None, **kwargs)
set_config(interactive=None, **kwargs)
get_config(name, print_console=True, interactive=None)
get_server_time()
```

1.1. API Server 5

```
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__()

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.
```

continues on next page

Table 5 – continued from previous page

<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	Does a zeroconf request to find a named AFL-automation server on the network.
<pre>sa_discover_server_by_name(service_name)</pre>	Does a zeroconf request to find a named AFL-automation server on the network.

#### \_\_init\_\_()

```
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.

1.1. API Server 7

```
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.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**

1.1. API Server 9

```
is_serialized(obj)
```

#### $AFL. automation. APIS erver. Queue Daemon. is \_serialized$

 ${\tt AFL.automation.APIServer.QueueDaemon.is\_serialized} (obj)$ 

#### 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

 ${\bf class} \ {\tt AFL.automation.APIS} erver. {\tt QueueDaemon.DataTrashcan}$ 

Bases: DataPacket

A DataPacket implementation for testing only that takes all its data and simply throws it away.

\_\_init\_\_()

#### **Methods**

init()	
add_array(*args, **kwargs)	Abstract method adding arrays that need special handling
clear()	
finalize()	
<pre>get(key[, default])</pre>	Retrieves the corresponding layout by the string key.
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.
reset()	Clears all transient data.
reset_sample()	
setdefault(k[,d])	
setupDefaults()	
transmit()	

continues on next page

Table 10 - continued from previous page

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_SAMPLE_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() → None. Remove all items from D.
```

get(key, default=None)

finalize()

Retrieves the corresponding layout by the string key.

When there isn't an exact match, all the existing keys in the layout map will be treated as a regex and map against the input key again. When there are multiple matches for the regex, an *ValueError* will be raised. Returns *None* if there isn't any match found.

#### **Parameters**

**key** – String key to query a layout.

#### Returns

Corresponding layout based on the query.

**items()**  $\rightarrow$  a set-like object providing a view on D's items

 $\textbf{keys}(\textbf{)} \rightarrow a \text{ set-like object providing a view on D's keys}$ 

 $pop(k[,d]) \rightarrow v$ , remove specified key and return the corresponding value.

If 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.

1.1. API Server 11

```
\begin{tabular}{l} \textbf{reset\_sample()} \\ \textbf{setdefault}(k[,d]) \rightarrow D.get(k,d), also set D[k]=d if k not in D \\ \textbf{setupDefaults()} \\ \textbf{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 \\ \textbf{values()} \rightarrow an object providing a view on D's values \\ \end{tabular}
```

#### AFL.automation.APIServer.QueueDaemon.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().

1.1. API Server 13

#### 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.

#### 1.2 Instrument

The Instrument module contains drivers and interfaces for various scientific instruments.

AFL.automation.instrument

#### 1.2.1 AFL.automation.instrument

#### **Modules**

FileCamera

NetworkCamera

#### AFL.automation.instrument.FileCamera

#### **Classes**

FileCamera()

1.2. Instrument 15

#### 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
```

#### AFL.automation.instrument.NetworkCamera

#### **Classes**

\_\_init\_\_()

collect(fname)

```
NetworkCamera(url)
```

#### AFL.automation.instrument.NetworkCamera.NetworkCamera

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

#### Methods

```
__init__(url)
camera_reset()

collect()

__init__(url)

camera_reset()

collect()
```

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

### 1.3 Loading

The Loading module provides functionality for loading and handling samples.

AFL.automation.loading

### 1.3.1 AFL.automation.loading

#### **Modules**

CetoniMultiPosValve	
ChemyxSyringePump	This is largely duplicated from the reference code provided by Chemyx.
DigitalOutPressureController	
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
MultiChannelRelay	
NE1kSyringePump	
PressureController	
PressureControllerAsPump	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
	continues on next page

#### Table 21 – continued from previous page

```
SyringePump

Tubing

UltimusVPressureController

ViciMultiposSelector
```

#### AFL.automation.loading.CetoniMultiPosValve

#### Classes

```
CetoniMultiPosValve(parentpump[, portlabels])

FlowSelector()
```

#### AFL.automation.loading.CetoniMultiPosValve.CetoniMultiPosValve

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

#### **Parameters**

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

#### Methods

init(parentpump[, portlabels])	connect to valve and query the number of positions
<pre>getPort([as_str])</pre>	query the current selected position
<pre>selectPort(port[, direction])</pre>	moves the selector to portnum

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

connect to valve and query the number of positions

#### **Parameters**

- use (baud baudrate to)

```
(portlabels - dict for smart port)

    naming

                                                                                              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.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)
                                    (portlabels - dict for smart port)

    naming

                                                                                             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,
```

1.3. Loading 19

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**

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

#### AFL.automation.loading.ChemyxSyringePump.getOpenPorts

AFL.automation.loading.ChemyxSyringePump.getOpenPorts()

#### AFL.automation.loading.ChemyxSyringePump.parsePortName

AFL.automation.loading.ChemyxSyringePump.parsePortName(portinfo)

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

#### **Classes**

```
ChemyxConnection(port, baudrate[, x, mode, ...])
ChemyxSyringePump(port, syringe_id_mm, ...)
SyringePump()
```

#### AFL.automation.loading.ChemyxSyringePump.ChemyxConnection

class AFL.automation.loading.ChemyxSyringePump.ChemyxConnection(port, baudrate, x=0, mode=0, verbose=False)

```
Bases: object
__init__(port, baudrate, x=0, mode=0, verbose=False)
```

#### **Methods**

```
__init__(port, baudrate[, x, mode, verbose])

addMode(command)

addX(command)
```

continues on next page

Table 27 – continued from previous page

```
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
```

#### **Methods**

baud = baudrate for connection

```
__init__(port, syringe_id_mm, syringe_volume)

blockUntilStatusStopped([pollingdelay])

dispense(volume[, block, delay])

emptySyringe()

Initializes and verifies connection to a Chemyx syringe pump.

This is a deprecated function from old serial logic.

continues on next page
```

Table 28 - continued from previous page

```
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)
```

stop()

Abort the current dispense/withdraw action.

#### 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.
```

```
withdraw(volume, block=True, delay=True)
     dispense(volume, block=True, delay=True)
     setRate(rate)
     getRate()
     emptySyringe()
     getLevel()
     setLevel(level)
     blockUntilStatusStopped(pollingdelay=0.2)
          This is a deprecated function from old serial logic. It should work, but do not use.
     getStatus()
          query the pump status and return whether the pump is moving or not (true if moving, false if stopped)
     getValueFromParams(search_key)
AFL.automation.loading.ChemyxSyringePump.getOpenPorts()
AFL.automation.loading.ChemyxSyringePump.parsePortName(portinfo)
     On macOS and Linux, selects only usbserial options and parses the 8 character serial number.
class AFL.automation.loading.ChemyxSyringePump.ChemyxConnection(port, baudrate, x=0, mode=0,
                                                                         verbose=False)
     __init__(port, baudrate, x=0, mode=0, verbose=False)
     openConnection()
     closeConnection()
     sendCommand(command)
     getResponse()
     startPump()
     stopPump()
     pausePump()
     restartPump()
     setUnits(units)
     setDiameter(diameter)
     setRate(rate)
     setVolume(volume)
     setDelay(delay)
     setTime(timer)
```

```
getParameterLimits()
getParameters()
getDisplacedVolume()
getElapsedTime()
getPumpStatus()
addMode(command)
addX(command)
```

#### AFL.automation.loading.DigitalOutPressureController

#### Classes

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

#### AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController

sure\_to\_v\_conv)

```
Bases: PressureController
__init__(digital_out, pressure_to_v_conv)
Initializes a DigitalOutPressureController
Params:
```

digital\_out (AFL.automation.DigitalOut): pressure\_to\_v\_conv (float): pressure units per volt

#### **Methods**

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

**\_\_init\_\_**(*digital\_out*, *pressure\_to\_v\_conv*)

Initializes a DigitalOutPressureController

Params:

digital\_out (AFL.automation.DigitalOut): pressure\_to\_v\_conv (float): pressure units per volt

set\_P(pressure)

#### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

#### dispenseRunning()

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

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

#### stop()

Abort the current timed dispense action.

timed\_dispense(dispense\_pressure, dispense\_time, block=True)

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

#### AFL.automation.loading.DigitalOutPressureController.PressureController

#### class AFL.automation.loading.DigitalOutPressureController.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.
timed_dispense(dispense_pressure, dispense_time)	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.

```
__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()

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

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

#### AFL.automation.loading.DoubleViciMultiposSelector.DoubleViciMultiposSelector

baudrate=9600, portlabels=None)

Bases: FlowSelector
\_\_init\_\_(port1, port2, baudrate=9600, portlabels=None)
connect to valve and query the number of positions

**Parameters** 

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

#### **Methods**

init(port1, port2[, baudrate, portlabels])	connect to valve and query the number of positions
<pre>getPort([as_str])</pre>	query the current selected position
<pre>selectPort(port[, direction])</pre>	moves the selector to portnum

\_\_init\_\_(port1, port2, baudrate=9600, portlabels=None) connect to valve and query the number of positions

#### **Parameters**

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

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

moves the selector to portnum

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

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

query the current selected position

#### AFL.automation.loading.DoubleViciMultiposSelector.FlowSelector

class AFL.automation.loading.DoubleViciMultiposSelector.FlowSelector

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

#### **Methods**

```
__init__()

getPort()

selectPort()
```

#### getPort()

#### selectPort()

#### AFL.automation.loading.DoubleViciMultiposSelector.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.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}

#### **Methods**

```
__init__(port[, baudrate, portlabels]) connect to valve and query the number of positions

getPort([as_str]) query the current selected position

selectPort(port[, direction, block]) moves the selector to portnum

sendCommand(cmd[, response, questionmarkOK,
...])
```

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

#### **Parameters**

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

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

moves the selector to portnum

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

#### **Parameters**

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

getPort(as str=False)

query the current selected position

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

 $\textbf{class} \ \texttt{AFL}. automation. loading. Double \texttt{ViciMultiposSelector}. \textbf{\textit{DoubleViciMultiposSelector}(port1, port1) and \texttt{\textit{Monthly of the property of the$ 

port2, baudrate=9600, portlabels=None)

\_\_init\_\_(port1, port2, baudrate=9600, portlabels=None)

connect to valve and query the number of positions

#### **Parameters**

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

selectPort(port, direction=None)

moves the selector to portnum

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

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

#### AFL.automation.loading.DummyPump

#### **Classes**

```
DummyPump()

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

SyringePump()
```

#### AFL.automation.loading.DummyPump.DummyPump

```
class AFL.automation.loading.DummyPump.DummyPump
Bases: SyringePump
__init__()
    Dummy pump for testing - does nothing, but boy does it look good doing it.
```

#### **Methods**

init()	Dummy pump for testing - does nothing, but boy does it look good doing it.
<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])	

```
__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)
```

continues on next page

```
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__()
     Methods
      __init__()
      dispense(volume[, block])
      emptySyringe()
      getRate(rate)
      setRate(rate)
      stop()
```

Table 41 – continued from previous page

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

#### Classes

```
FlowSelector()
```

# AFL.automation.loading.FlowSelector.FlowSelector

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

```
__init__()
getPort()

getPort()

getPort()

selectPort()

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

# 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)

toggleChannels(channels)
```

```
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

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.

```
__init__(port, syringe_id_mm, syringe_volume)

blockUntilStatusStopped([pollingdelay])

dispense(volume[, block, delay])

emptySyringe()

getRate()

getStatus()

setRate(rate)

setRate(rate)

stop()

withdraw(volume[, block, delay])

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

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

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

Abort the current dispense/withdraw action.
```

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)

stop()

```
setRate(rate)
     getRate()
     emptySyringe()
     blockUntilStatusStopped(pollingdelay=0.2)
     getStatus()
          query the pump status and return a tuple of the status character, infused volume, and withdrawn volume)
AFL.automation.loading.NE1kSyringePump.SerialDevice
class AFL.automation.loading.NE1kSyringePump.SerialDevice(port, baudrate=19200, timeout=0.5,
                                                                 raw_writes=False)
     Bases: object
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     Methods
      __init__(port[, baudrate, timeout, raw_writes])
      sendCommand(cmd[, response, questionmarkOK,
      ...])
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
AFL.automation.loading.NE1kSyringePump.SyringePump
class AFL.automation.loading.NE1kSyringePump.SyringePump
     Bases: object
     __init__()
     Methods
      __init__()
      dispense(volume[, block])
      emptySyringe()
      getRate(rate)
```

continues on next page

# Table 49 – continued from previous page

```
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.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

```
\textbf{class} \ AFL. automation. loading. Pressure Controller As Pump. \textbf{Pressure Controller}. Pressure Controller As Pump. \textbf{Pressure Controller}. The property of the pressure of the pressure controller as Pump. \textbf{Pressure Controller}. The pressure controller as Pump. \textbf{Pressure C
```

```
dis-
pense_pressure=3.5,
im-
plied_flow_rate=50)
```

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

\_\_init\_\_(port, baudrate=19200, timeout=0.5, raw\_writes=False)

```
class AFL.automation.loading.PressureControllerAsPump.SerialDevice(port, baudrate=19200, timeout=0.5, raw\_writes=False)

Bases: object
```

```
__init__(port[, baudrate, timeout, raw_writes])
 sendCommand(cmd[, response, questionmarkOK,
 ...])
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

# AFL.automation.loading.PressureControllerAsPump.SyringePump

```
\textbf{class} \ \texttt{AFL}. automation. loading. Pressure Controller \texttt{AsPump}. \textbf{Syringe Pump}
       Bases: object
       __init__()
```

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

#### **Methods**

```
__init__()
dispense(volume[, block])
emptySyringe()
getRate(rate)
setRate(rate)
stop()
withdraw(volume[, block])
```

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

class AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump(pressure\_controller, dis-

 $pense\_pressure=3.5$ ,

plied\_flow\_rate=50)

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

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

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

withdraw(volume, block=True, delay=True)

dispense(volume, block=True, delay=True)

setRate(rate)

getRate()

emptySyringe()

blockUntilStatusStopped(pollingdelay=0.2)

getStatus()

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

# AFL.automation.loading.SainSmartRelay

#### **Module Attributes**

```
usbrelay
```

#### AFL.automation.loading.SainSmartRelay.usbrelay

```
AFL.automation.loading.SainSmartRelay.usbrelay = [[b':FE0100200000FF\r\n', b':FE01000000010F1\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':FE050005F00F8\r\n'], [b':FE050005FF00F9\r\n'], [b':FE0500060000F7\r\n', b':FE050006FF00F8\r\n'], [b':FE0500070000F6\r\n', b':FE050007F00F7\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':FE05000CFF00F2\r\n'], [b':FE05000DF00F1\r\n', b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFF00FF\r\n'], [b':FE05000EFFT00FF\r\n'], [b':FE05000EFFTE3\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)
```

"

#### **Classes**

```
MultiChannelRelay()

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

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

# AFL.automation.loading.SainSmartRelay.MultiChannelRelay

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

#### **Methods**

```
__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
```

```
__init__(relaylabels, serial_port[, timeout])
                                                           Init connection to a SainSmart 16-channel USB relay
                                                           module.
       getChannels([asid])
                                                           Read the current state of all channels
       sendCommand(cmd[, response, questionmarkOK,
       setAllChannelsOff()
       setChannels(channels)
                                                           Write a value (True, False) to the channels specified
                                                           in channels
       toggleChannels(channels)
     __init__(relaylabels, serial_port, timeout=0.5)
           Init connection to a SainSmart 16-channel USB relay module.
           Params: relaylabels (dict):
               mapping of port id to load name, e.g. {0:'arm_up',1:'arm_down'}
           serial_port (str):
               port to connect to
     setAllChannelsOff()
     setChannels(channels)
           Write a value (True, False) to the channels specified in channels
           Parameters: channels (dict):
               dict of channels, keys as either str (name) or int (id) to write to, vals as the value to write
     getChannels(asid=False)
           Read the current state of all channels
           Parameters: asid (bool,default false): Dict keys should simply be the id, not the name.
           Returns: (dict) key:value mappings of state.
     toggleChannels(channels)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
AFL.automation.loading.SainSmartRelay.SerialDevice
class AFL.automation.loading.SainSmartRelay.SerialDevice(port, baudrate=19200, timeout=0.5,
                                                                      raw_writes=False)
     Bases: object
     __init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

is:

```
__init__(port[, baudrate, timeout, raw_writes])
      sendCommand(cmd[, response, questionmarkOK,
      ...])
     __init__(port, baudrate=19200, timeout=0.5, raw writes=False)
     sendCommand(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)
class AFL.automation.loading.SainSmartRelay.SainSmartRelay(relaylabels, serial_port, timeout=0.5)
     __init__(relaylabels, serial_port, timeout=0.5)
          Init connection to a SainSmart 16-channel USB relay module.
          Params: relaylabels (dict):
              mapping of port id to load name, e.g. {0:'arm_up',1:'arm_down'}
          serial_port (str):
              port to connect to
     setAllChannelsOff()
     setChannels(channels)
          Write a value (True, False) to the channels specified in channels
          Parameters: channels (dict):
              dict of channels, keys as either str (name) or int (id) to write to, vals as the value to write
     getChannels(asid=False)
          Read the current state of all channels
          Parameters: asid (bool,default false): Dict keys should simply be the id, not the name.
          Returns: (dict) key:value mappings of state.
     toggleChannels(channels)
AFL.automation.loading.SainSmartRelay.usbrelay = [[b':FE0100200000FF\r\n',
b':FE0100000010F1\r\n'], [b':FE050000000FD\r\n', b':FE050000FF00FE\r\n'],
[b':FE0500010000FC\r\n', b':FE050001FF00FD\r\n'], [b':FE0500020000FB\r\n',
b':FE050002FF00FC\r\n'], [b':FE0500030000FA\r\n', b':FE050003FF00FB\r\n'],
[b':FE0500040000F9\r\n', b':FE050004FF00FA\r\n'], [b':FE0500050000F8\r\n',
b':FE050005FF00F9\r\n'], [b':FE0500060000F7\r\n', b':FE050006FF00F8\r\n'],
[b':FE0500070000F6\r\n', b':FE050007FF00F7\r\n'], [b':FE0500080000F5\r\n',
b':FE050008FF00F6\r\n'], [b':FE0500090000F4\r\n', b':FE050009FF00F5\r\n'],
[b':FE05000A0000F3\r\n', b':FE05000AFF00F4\r\n'], [b':FE05000B0000F2\r\n',
b':FE05000BFF00F3\r\n'], [b':FE05000C0000F1\r\n', b':FE05000CFF00F2\r\n'],
[b':FE05000D0000F0\r\n', b':FE05000DFF00F1\r\n'], [b':FE05000E0000FF\r\n',
b':FE05000EFF00F0\r\n'], [b':FE05000F0000FE\r\n', b':FE05000FF00FF\r\n'],
[b':FE0F00000010020000E1\r\n', b':FE0F0000001002FFFFE3\r\n']]
     "" Sainsmart 16-Channel 9-36v USB Relay Module (CH341 chip)(sku# 101-70-208) 1. Requires CH341 Win-
     dows driver installed (see http://wiki.sainsmart.com/index.php/101-70-208) 2. The hex table provided by Sains-
     mart requires converting to ASCII chars (see 'usbrelay' below) 3. Format of 'usbrelay' two-dimensional array
```

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

while the 1st index selects the array row.

#### Example...

 $status = usbrelay[0][1] \# row-0 (status) stat\_ret = usbrelay[0][0] \# row-0 (status return) ch\_1\_on = usbrelay[1][1] \# row-1 (chan-1 off) ch\_1\_off = usbrelay[1][0] \# row-1 (chan-1 off) ch\_16\_on = usbrelay[16][1] \# row-16 (chan-16 on) ch\_16\_off = usbrelay[16][0] \# row-16 (chan-16 off) all\_on = usbrelay[17][1] \# row-17 (all on) all\_off = usbrelay[17][0] \# row-17 (all off)$ 

٠,,,,

# AFL.automation.loading.SampleCell

#### Classes

SampleCell()

# AFL.automation.loading.SampleCell.SampleCell

class AFL.automation.loading.SampleCell.SampleCell

Bases: object
\_\_init\_\_()

#### **Methods**

```
__init__()
```

loadSample()

# loadSample()

class AFL.automation.loading.SampleCell.SampleCell

loadSample()

# AFL.automation.loading.Sensor

#### **Classes**

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

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

Sensor([address, channel])
```

# AFL.automation.loading.Sensor.DummySensor1

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

Bases: Thread

**\_\_init\_\_**(period=2, minval=-5, maxval=5)

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

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

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

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

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

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

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

#### **Methods**

init([period, minval, maxval])	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

**\_\_init\_\_**(*period*=2, *minval*=-5, *maxval*=5)

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

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

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

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

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

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

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

#### read()

#### terminate()

#### run()

Method representing the thread's activity.

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

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

# property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

# isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

## setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

# start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

# AFL.automation.loading.Sensor.DummySensor2

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

Bases: Thread

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

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

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

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

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

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

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

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

init([period, hi_value, lo_time, hi_time])	This constructor should always be called with keyword arguments.
<pre>driver_status()</pre>	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
read()	
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.
setName(name)	Set the name string for this thread.
start()	Start the thread's activity.
terminate()	

#### **Attributes**

daemon	A boolean value indicating whether this thread is a daemon thread.
ident	Thread identifier of this thread or None if it has not been started.
name	A string used for identification purposes only.
native_id	Native integral thread ID of this thread, or None if it has not been started.

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

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

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

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

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

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

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

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

```
driver_status()
read()
terminate()
run()
```

Method representing the thread's activity.

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

#### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

#### getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

#### property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

#### is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

```
setDaemon(daemonic)
          Set whether this thread is a daemon.
          This method is deprecated, use the .daemon property instead.
     setName(name)
          Set the name string for this thread.
          This method is deprecated, use the name attribute instead.
     start()
          Start the thread's activity.
          It must be called at most once per thread object. It arranges for the object's run() method to be invoked in
          a separate thread of control.
          This method will raise a RuntimeError if called more than once on the same thread object.
AFL.automation.loading.Sensor.Sensor
class AFL.automation.loading.Sensor.Sensor(address=1, channel=0)
     Bases: object
     __init__(address=1, channel=0)
     Methods
       __init__([address, channel])
       calibrate()
       read()
     __init__(address=1, channel=0)
     calibrate()
     read()
class AFL.automation.loading.Sensor.Sensor(address=1, channel=0)
     __init__(address=1, channel=0)
     calibrate()
     read()
class AFL.automation.loading.Sensor.DummySensor1(period=2, minval=-5, maxval=5)
     __init__(period=2, minval=-5, maxval=5)
          This constructor should always be called with keyword arguments. Arguments are:
```

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

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

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

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, ...])
```

continues on next page

# Table 69 – continued from previous page

```
StopLoadCBv1(poll, period, load_client[, ...])
StopLoadCBv2(poll, period[, load_client, ...])
```

# AFL.automation.loading.SensorCallbackThread.LoaderCommunication

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

#### **Methods**

```
__init__([load_client, load_object])

getServerState()

stopLoad()

__init__(load_client=None, load_object=None)

getServerState()

stopLoad()
```

# AFL.automation.loading.SensorCallbackThread.SensorCallbackThread

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

```
Bases: Thread
```

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

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

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

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

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

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

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

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

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.

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()	

continues on next page

# Table 73 - continued from previous page

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)
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()

 $\begin{tabular}{ll} \textbf{class} & \textbf{AFL}. \textbf{automation.} \textbf{loading.} \textbf{SensorCallbackThread.} \textbf{SimpleThresholdCB} (poll, period, window=5, threshold=1) \end{tabular}$ 

```
__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.

1.3. Loading 69

```
process_signal()
```

```
__init__(load_client=None, load_object=None)
getServerState()
stopLoad()
```

## AFL.automation.loading.SensorPollingThread

#### **Classes**

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

## AFL.automation.loading.SensorPollingThread.SensorPollingThread

 $\textbf{class} \ \texttt{AFL.automation.loading.SensorPollingThread}. \textbf{SensorPollingThread} (\textit{sensor}, \textit{period} = 0.1, \texttt{automation.loading.SensorPollingThread}) (\textit{sensor}, \textit{perio$ 

callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

Bases: Thread

\_\_init\_\_(sensor, period=0.1, callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

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

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

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

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

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

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

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

#### **Methods**

init(sensor[, period, callback,])	This constructor should always be called with keyword arguments.
alive()	
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.

continues on next page

Table 80 - continued from previous page

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()

1.3. Loading 71

#### run()

Method representing the thread's activity.

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

### property daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when only daemon threads are left.

## getName()

Return a string used for identification purposes only.

This method is deprecated, use the name attribute instead.

## property ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

#### isDaemon()

Return whether this thread is a daemon.

This method is deprecated, use the daemon attribute instead.

## is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. See also the module function enumerate().

#### join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call is\_alive() after join() to decide whether a timeout happened – if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

#### property name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

#### property native\_id

Native integral thread ID of this thread, or None if it has not been started.

This is a non-negative integer. See the get\_native\_id() function. This represents the Thread ID as reported by the kernel.

#### setDaemon(daemonic)

Set whether this thread is a daemon.

This method is deprecated, use the .daemon property instead.

#### setName(name)

Set the name string for this thread.

This method is deprecated, use the name attribute instead.

#### start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

```
class AFL.automation.loading.SensorPollingThread.SensorPollingThread(sensor, period=0.1,
```

callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

\_\_init\_\_(sensor, period=0.1, callback=None, hv\_pipe=None, window=None, filename=None, daemon=True, data=None)

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

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

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

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

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

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

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

read()

read\_load\_buffer()

reset\_load\_buffer()

terminate()

alive()

1.3. Loading 73

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

```
\textbf{exception} \hspace{0.1cm} \textbf{AFL}. automation. loading. Serial Device. \textbf{Serial Comms Exception} \\
```

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()
```

1.3. Loading 75

## 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.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
<pre>dispenseRunning()</pre>	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.

1.3. Loading 77

## AFL.automation.loading.UltimusVPressureController.UltimusVPressureController

 $\textbf{class} \ \texttt{AFL}. \textbf{automation.} loading. \textbf{Ultimus VP} ressure \texttt{Controller.} \textbf{Ultimus VP} \textbf{ressure Controller.} \textbf{Ultimus VP} \textbf{ressure Controller.} \textbf{VP} \textbf{$ 

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.

#### dispenseRunning()

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

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

```
stop()
```

Abort the current timed dispense action.

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

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

 $\textbf{class} \ \texttt{AFL}. \textbf{automation.} loading. \textbf{UltimusVPressureController.} \textbf{UltimusVPressureController} (\textit{port}, \textit{port}, \textit{po$ 

baud=115200)

### AFL.automation.loading.ViciMultiposSelector

## **Classes**

```
FlowSelector()

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

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

## AFL.automation.loading.ViciMultiposSelector.FlowSelector

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

1.3. Loading 79

### **Methods**

```
__init__()

getPort()

selectPort()
```

```
getPort()
```

selectPort()

## AFL.automation.loading.ViciMultiposSelector.SerialDevice

```
\begin{tabular}{ll} \textbf{class} & \textbf{AFL.automation.loading.ViciMultiposSelector.SerialDevice}(port,baudrate=19200,\\ & timeout=0.5,raw\_writes=False) \end{tabular}
```

```
Bases: object
__init__(port, baudrate=19200, timeout=0.5, raw_writes=False)
```

#### **Methods**

```
__init__(port[, baudrate, timeout, raw_writes])

sendCommand(cmd[, response, questionmarkOK,
...])
```

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

**sendCommand**(*cmd*, *response=True*, *questionmarkOK=False*, *timeout=-1*, *debug=False*)

## AFL.automation.loading.ViciMultiposSelector.ViciMultiposSelector

```
Bases: SerialDevice, FlowSelector
```

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

#### **Parameters**

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

### **Methods**

```
__init__(port[, baudrate, portlabels]) connect to valve and query the number of positions

getPort([as_str]) query the current selected position

selectPort(port[, direction, block]) moves the selector to portnum

sendCommand(cmd[, response, questionmarkOK,
...])
```

```
__init__(port, baudrate=9600, portlabels=None)
```

connect to valve and query the number of positions

#### **Parameters**

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

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

moves the selector to portnum

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

#### **Parameters**

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

getPort(as str=False)

query the current selected position

**sendCommand**(cmd, response=True, questionmarkOK=False, timeout=-1, debug=False)

\_\_init\_\_(port, baudrate=9600, portlabels=None)

connect to valve and query the number of positions

#### **Parameters**

- 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}

1.3. Loading 81

```
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

# 1.4 Prepare

The Prepare module provides functionality for sample preparation and processing.

# 1.5 Sample

The Sample module provides functionality for experiment orchestration 'Sample Servers'

```
AFL.automation.sample
AFL.automation.sample_env
```

# 1.5.1 AFL.automation.sample

## **Modules**

## 1.5.2 AFL.automation.sample\_env

### **Modules**



## 1.6 Shared

The Shared module provides common utilities and functionality used across the AFL-automation framework.

```
AFL.automation.shared
```

## 1.6.1 AFL.automation.shared

#### **Modules**

DataLabelerWidget

DatasetWidget

DiffractionLabeler

MutableQueue

PersistentConfig

ServerDiscovery

exceptions

serialization

widgetui

## AFL.automation.shared.DataLabelerWidget

#### **Functions**

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

## AFL.automation.shared.DataLabelerWidget.sqrt

AFL.automation.shared.DataLabelerWidget. $sqrt(x,/(Positional-only\ parameter\ separator\ (PEP\ 570)))$ Return the square root of x.

## **Classes**

```
DataLabelerModel(dataset)

DataLabelerView()

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

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

Encode categorical features as an integer array.
```

## AFL.automation.shared.DataLabelerWidget.DataLabelerModel

### **Methods**

```
__init__(dataset)
      get_peaks(model[, qstar, max_order])
      init_models()
      ordinal_phase_labels()
     __init__(dataset: Dataset)
     ordinal_phase_labels()
     init_models()
     get_peaks(model, qstar=None, max_order=4)
AFL.automation.shared.DataLabelerWidget.DataLabelerView
class AFL.automation.shared.DataLabelerWidget.DataLabelerView
     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_composition_colors(colors)
      update_plot(x, y, composition)
     __init__()
     update_plot(x, y, composition)
     remove_vertical_lines()
     add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
     update_composition_colors(colors)
     run(x, y, all\_compositions, composition, models, ternary, components)
```

## AFL.automation.shared.DataLabelerWidget.DataLabelerWidget

```
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)
```

Bases: object

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

#### **Parameters**

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

#### **Methods**

```
__init__(input_dataset, sas_variable, ...[, ...])

change_model_callback(data)

change_norder_callback(figure, location, click)

composition_click_callback(figure, location, ...)

draw_peaks(peaks)

goto_callback(click)

label(label)

next_button_callback(click)

prev_button_callback(click)

run()
```

continues on next page

## Table 107 – continued from previous page

```
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  $unknown\_value$  or  $encoded\_missing\_value$  to an integer will increase the number of unique integer codes by one each. This can result in up to  $max\_categories + 2$  integer codes.

Added in version 1.3: Read more in the User Guide.

### categories\_

The categories of each feature determined during fit (in order of the features in X and corresponding with the output of transform). This does not include categories that weren't seen during fit.

```
Type
```

list of arrays

## n\_features\_in\_

Number of features seen during fit.

Added in version 1.0.

## Type

int

#### feature\_names\_in\_

Names of features seen during fit. Defined only when *X* has feature names that are all strings.

Added in version 1.0.

## **Type**

ndarray of shape (n\_features\_in\_,)

### infrequent\_categories\_

Defined only if infrequent categories are enabled by setting *min\_frequency* or *max\_categories* to a non-default value. *infrequent\_categories\_[i]* are the infrequent categories for feature *i*. If the feature *i* has no infrequent categories *infrequent\_categories\_[i]* is None.

Added in version 1.3.

## Type

list of ndarray

# See also

## OneHotEncoder

Performs a one-hot encoding of categorical features. This encoding is suitable for low to medium cardinality categorical variables, both in supervised and unsupervised settings.

## TargetEncoder

Encodes categorical features using supervised signal in a classification or regression pipeline. This encoding is typically suitable for high cardinality categorical variables.

## LabelEncoder

Encodes target labels with values between 0 and n\_classes-1.

## **Notes**

With a high proportion of *nan* values, inferring categories becomes slow with Python versions before 3.10. The handling of *nan* values was improved from Python 3.10 onwards, (c.f. bpo-43475).

#### **Examples**

Given a dataset with two features, we let the encoder find the unique values per feature and transform the data to an ordinal encoding.

By default, OrdinalEncoder is lenient towards missing values by propagating them.

You can use the parameter *encoded\_missing\_value* to encode missing values.

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.],
       ſ1.],
       [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**

```
__init__(*, categories='auto', dtype=<class 'numpy.float64'>, handle_unknown='error', unknown_value=None, encoded_missing_value=nan, min_frequency=None, max_categories=None)
```

#### fit(X, y=None)

Fit the OrdinalEncoder to X.

#### **Parameters**

- **X** (array-like of shape (n\_samples, n\_features)) The data to determine the categories of each feature.
- y (None) Ignored. This parameter exists only for compatibility with Pipeline.

#### Returns

self – Fitted encoder.

## Return type

object

## transform(X)

Transform X to ordinal codes.

#### **Parameters**

 $\mathbf{X}$  (array-like of shape (n\_samples, n\_features)) - The data to encode.

#### **Returns**

 $X_out$  – Transformed input.

## Return type

ndarray of shape (n\_samples, n\_features)

## $inverse\_transform(X)$

Convert the data back to the original representation.

#### **Parameters**

 $\boldsymbol{\mathtt{X}}$  (array-like of shape (n\_samples, n\_encoded\_features)) – The transformed data.

#### Returns

**X\_tr** – Inverse transformed array.

## Return type

ndarray of shape (n\_samples, n\_features)

## fit\_transform(X, y=None, \*\*fit\_params)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit\_params* and returns a transformed version of *X*.

#### **Parameters**

- X (array-like of shape (n\_samples, n\_features)) Input samples.
- y (array-like of shape (n\_samples,) or (n\_samples, n\_outputs), default=None) Target values (None for unsupervised transformations).
- \*\*fit\_params (dict) Additional fit parameters.

#### Returns

**X\_new** – Transformed array.

### Return type

ndarray array of shape (n\_samples, n\_features\_new)

### get\_feature\_names\_out(input\_features=None)

Get output feature names for transformation.

#### **Parameters**

input\_features (array-like of str or None, default=None) - Input features.

- If *input\_features* is *None*, then *feature\_names\_in\_* is used as feature names in. If *feature\_names\_in\_* is not defined, then the following input feature names are generated: ["x0", "x1", ..., "x(n\_features\_in\_ 1)"].
- If *input\_features* is an array-like, then *input\_features* must match *feature\_names\_in\_* if *feature\_names\_in\_* is defined.

#### Returns

**feature\_names\_out** – Same as input features.

#### **Return type**

ndarray of str objects

## get\_metadata\_routing()

Get metadata routing of this object.

Please check User Guide on how the routing mechanism works.

### Returns

**routing** – A MetadataRequest encapsulating routing information.

## Return type

MetadataRequest

## get\_params(deep=True)

Get parameters for this estimator.

## **Parameters**

**deep** (*bool*, *default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

#### Returns

**params** – Parameter names mapped to their values.

## Return type

dict

## property infrequent\_categories\_

Infrequent categories for each feature.

set\_output(\*(Keyword-only parameters separator (PEP 3102)), transform=None)

Set output container.

See sphx\_glr\_auto\_examples\_miscellaneous\_plot\_set\_output.py for an example on how to use the API.

#### **Parameters**

**transform** ({"default", "pandas", "polars"}, default=None) - Configure output of *transform* and *fit\_transform*.

- "default": Default output format of a transformer
- "pandas": DataFrame output
- "polars": Polars output
- None: Transform configuration is unchanged

Added in version 1.4: "polars" option was added.

#### Returns

**self** – Estimator instance.

### **Return type**

estimator instance

```
set_params(**params)
```

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as Pipeline). The latter have parameters of the form <component>\_\_<parameter> so that it's possible to update each component of a nested object.

## **Parameters**

```
**params (dict) – Estimator parameters.
```

#### Returns

**self** – Estimator instance.

#### **Return type**

estimator instance

class AFL.automation.shared.DataLabelerWidget.DataLabelerWidget(input\_dataset: Dataset,

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

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

#### **Parameters**

• **dataset** (xr.Dataset) – Dataset from AFL

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

```
next_button_callback(click)
     prev_button_callback(click)
     goto_callback(click)
     composition_click_callback(figure, location, click)
     update_plot()
     draw_peaks(peaks)
     change_qstar_callback(figure, location, click)
     change_model_callback(data)
     change_norder_callback(data)
     label(label)
     run()
class AFL.automation.shared.DataLabelerWidget.DataLabelerModel(dataset: Dataset)
     __init__(dataset: Dataset)
     ordinal_phase_labels()
     init_models()
     get_peaks(model, qstar=None, max order=4)
class AFL.automation.shared.DataLabelerWidget.DataLabelerView
     __init__()
     update_plot(x, y, composition)
     remove_vertical_lines()
     add_vertical_line(x, y0=0, y1=128, row=1, col=1, line_kw=None)
     update_composition_colors(colors)
     run(x, y, all\_compositions, composition, models, ternary, components)
```

## AFL.automation.shared.DatasetWidget

#### **Classes**

## AFL.automation.shared.DatasetWidget.DatasetWidget

Bases: object

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

Interactive widget for viewing compositionally varying scattering data

### **Parameters**

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

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

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

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

```
```python -DatasetWidget(ds) (widget =)widget.run()``` -
```

### **Methods**

```
__init__(dataset[, sample_dim, ...])
  Interactive widget for viewing compositionally vary-
  ing scattering data
apply_isel(*args)
apply_sel(*args)
combine_vars(*args)
composition_click_callback(figure, location,
extract_var(*args)
get_comps()
goto_callback(*args)
initialize_plots(*args)
next_button_callback(*args)
prev_button_callback(*args)
reset_dataset(*args)
run()
update_colors(*args)
update_composition_plot()
update_dropdowns(*args)
update_extract_coords(change)
update_plots()
update_sample_dim(*args)
update_scattering_plot()
```

\_\_init\_\_(dataset: Dataset, sample\_dim: str = 'sample', scatt\_variables: List[str] | None = None, comps\_variable: str | None = None, comps\_color\_variable: str | None = None, xmin: float = 0.001, xmax: float = 1.0)

Interactive widget for viewing compositionally varying scattering data

#### **Parameters**

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

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

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

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

```
• Usage
            • ----
            • ```python -
            • DatasetWidget(ds) (widget =)
            widget.run()
            • ``` _
next_button_callback(*args)
prev_button_callback(*args)
goto_callback(*args)
composition_click_callback(figure, location, click)
update_composition_plot()
update_scattering_plot()
update_plots()
get_comps()
initialize_plots(*args)
update_colors(*args)
apply_sel(*args)
```

```
apply_isel(*args)
extract_var(*args)
combine_vars(*args)
reset_dataset(*args)
update_dropdowns(*args)
update_sample_dim(*args)
update_extract_coords(change)
run()
```

## AFL.automation.shared.DatasetWidget.DatasetWidget\_Model

#### **Methods**

```
__init__(dataset, sample_dim)

apply_isel(kw)

apply_sel(kw)

combine_vars(combined_var, to_combine_vars)

extract_var(extract_from_var, 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**

```
dataset

__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_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 \mid 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])
```

continues on next page

## Table 114 - continued from previous page

```
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_dropdowns()

init_inputs()
```

#### AFL.automation.shared.DatasetWidget.defaultdict

### class AFL.automation.shared.DatasetWidget.defaultdict

```
Bases: dict
```

run()

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()

fromkeys(iterable[, value])

get(key[, default])

Return the value for key if key is in the dictionary, else default.

continues on next page
```

Table 115 - continued from previous page

<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

init(*args, **kwargs)	
$\textbf{clear()} \rightarrow \text{None. Remove all items from D.}$	
$copy() \rightarrow a$ shallow copy of D.	
default_factory  Factory for default value called bymissing	().
classmethod fromkeys(iterable, value=None,/) Create a new dictionary with keys from iterable	and values set to value.
<pre>get(key, default=None, /)     Return the value for key if key is in the dictionary</pre>	ry, else default.

Factory for default value called by missing ().

 $\textbf{items}(\textbf{)} \rightarrow \textbf{a set-like object providing a view on D's items}$ 

**keys**()  $\rightarrow$  a set-like object providing a view on D's keys

 $pop(k[,d]) \rightarrow v$ , remove specified key and return the corresponding value.

If the key is not found, return the default if given; otherwise, raise a KeyError.

## popitem()

Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order. Raises KeyError if the dict is empty.

## setdefault(key, default=None, /)

Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default.

**update**( $[E, ]^{**F}$ )  $\rightarrow$  None. Update D from dict/iterable E and F.

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

```
values() \rightarrow an object providing a view on D's values
```

```
class AFL.automation.shared.DatasetWidget.DatasetWidget(dataset: Dataset, sample_dim: str = sample', scatt_variables: List[str] \mid None = None, comps_variable: str \mid None = None, comps_variable: str \mid None = None, stat_variable: str \mid No
```

```
__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
- ullet 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)
```

#### AFL.automation.shared.DiffractionLabeler

### **Functions**

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

## AFL.automation.shared.DiffractionLabeler.sqrt

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

#### **Classes**

```
DiffractionLabeler(saxs_data, ...)

DiffractionLabelerModel(saxs_data, ...)

DiffractionLabelerView()

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

Encode categorical features as an integer array.
```

### AFL.automation.shared.DiffractionLabeler.DiffractionLabeler

 $\textbf{class} \ \, \textbf{AFL}. \textbf{automation.shared.DiffractionLabeler.DiffractionLabeler} (saxs\_data, composition\_data, \\ possible\_phase\_labels)$ 

Bases: object
\_\_init\_\_(saxs\_data, composition\_data, possible\_phase\_labels)

### **Methods**

```
__init__(saxs_data, composition_data, ...)
 change_model_callback(data)
 change_norder_callback(data)
 change_qstar_callback(figure, location, click)
 draw_peaks(peaks)
 goto_callback(click)
 label(label)
 next_button_callback(click)
 prev_button_callback(click)
 run()
 ternary_click_callback(figure, location, click)
 update_plot()
__init__(saxs_data, composition_data, possible_phase_labels)
next_button_callback(click)
prev_button_callback(click)
goto_callback(click)
ternary_click_callback(figure, location, click)
update_plot()
draw_peaks(peaks)
{\bf change\_qstar\_callback} ({\it figure, location, click})
change_model_callback(data)
change_norder_callback(data)
label(label)
run()
```

AFL.automation.shared.DiffractionLabeler.DiffractionLabelerModel

```
Bases: object
__init__(saxs_data, composition_data, possible_phase_labels)
```

# **Methods**

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

get_peaks(model[, qstar, max_order])

init_models()

ordinal_phase_labels()

__init__(saxs_data, composition_data, possible_phase_labels)
```

```
ordinal_phase_labels()
init_models()
get_peaks(model, qstar=None, max_order=4)
```

# AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView

 ${\bf class} \ \, {\tt AFL.automation.shared.DiffractionLabeler.DiffractionLabelerView}$ 

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

# **Methods**

update\_plot(x, y, composition)

```
__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)
```

```
remove_vertical_lines()  \begin{subarray}{l} 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) \\ \end{subarray}
```

# AFL.automation.shared.DiffractionLabeler.OrdinalEncoder

Bases: OneToOneFeatureMixin, \_BaseEncoder

Encode categorical features as an integer array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are converted to ordinal integers. This results in a single column of integers (0 to n\_categories - 1) per feature.

Read more in the User Guide. For a comparison of different encoders, refer to: sphx\_glr\_auto\_examples\_preprocessing\_plot\_target\_encoder.py.

Added in version 0.20.

### **Parameters**

- categories ('auto' or a list of array-like, default='auto') Categories (unique values) per feature:
  - 'auto': Determine categories automatically from the training data.
  - list: categories[i] holds the categories expected in the ith column. The passed categories should not mix strings and numeric values, and should be sorted in case of numeric values.

The used categories can be found in the categories\_ attribute.

- **dtype** (number type, default=np.float64) Desired dtype of output.
- handle\_unknown ({'error', 'use\_encoded\_value'}, default='error') When set to 'error' an error will be raised in case an unknown categorical feature is present during transform. When set to 'use\_encoded\_value', the encoded value of unknown categories will be set to the value given for the parameter unknown\_value. In inverse\_transform(), an unknown category will be denoted as None.

Added in version 0.24.

• unknown\_value (int or np.nan, default=None) — When the parameter handle\_unknown is set to 'use\_encoded\_value', this parameter is required and will set the encoded value of unknown categories. It has to be distinct from the values used to encode any of the categories in fit. If set to np.nan, the dtype parameter must be a float dtype.

Added in version 0.24.

• **encoded\_missing\_value** (*int or np.nan*, *default=np.nan*) – Encoded value of missing categories. If set to *np.nan*, then the *dtype* parameter must be a float dtype.

Added in version 1.1.

- min\_frequency (int or float, default=None) Specifies the minimum frequency below which a category will be considered infrequent.
  - If *int*, categories with a smaller cardinality will be considered infrequent.
  - If float, categories with a smaller cardinality than min\_frequency \* n\_samples will be considered infrequent.

Added in version 1.3: Read more in the User Guide.

• max\_categories (int, default=None) – Specifies an upper limit to the number of output categories for each input feature when considering infrequent categories. If there are infrequent categories, max\_categories includes the category representing the infrequent categories along with the frequent categories. If None, there is no limit to the number of output features.

max\_categories do **not** take into account missing or unknown categories. Setting unknown\_value or encoded\_missing\_value to an integer will increase the number of unique integer codes by one each. This can result in up to max\_categories + 2 integer codes.

Added in version 1.3: Read more in the User Guide.

# categories\_

The categories of each feature determined during fit (in order of the features in X and corresponding with the output of transform). This does not include categories that weren't seen during fit.

### **Type**

list of arrays

## n\_features\_in\_

Number of features seen during fit.

Added in version 1.0.

# Type

int

# feature\_names\_in\_

Names of features seen during fit. Defined only when *X* has feature names that are all strings.

Added in version 1.0.

### **Type**

ndarray of shape (n\_features\_in\_,)

# infrequent\_categories\_

Defined only if infrequent categories are enabled by setting *min\_frequency* or *max\_categories* to a non-default value. *infrequent\_categories\_[i]* are the infrequent categories for feature *i*. If the feature *i* has no infrequent categories *infrequent\_categories\_[i]* is None.

Added in version 1.3.

# **Type**

list of ndarray

### See also

## OneHotEncoder

Performs a one-hot encoding of categorical features. This encoding is suitable for low to medium cardinality categorical variables, both in supervised and unsupervised settings.

### TargetEncoder

Encodes categorical features using supervised signal in a classification or regression pipeline. This encoding is typically suitable for high cardinality categorical variables.

### LabelEncoder

Encodes target labels with values between 0 and n\_classes-1.

### **Notes**

With a high proportion of *nan* values, inferring categories becomes slow with Python versions before 3.10. The handling of *nan* values was improved from Python 3.10 onwards, (c.f. bpo-43475).

# **Examples**

Given a dataset with two features, we let the encoder find the unique values per feature and transform the data to an ordinal encoding.

By default, *OrdinalEncoder* is lenient towards missing values by propagating them.

You can use the parameter *encoded\_missing\_value* to encode missing values.

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**

```
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(key[, default])</pre>	Retrieves the corresponding layout by the string key.
<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.
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(key, default=None)

Retrieves the corresponding layout by the string key.

When there isn't an exact match, all the existing keys in the layout map will be treated as a regex and map against the input key again. When there are multiple matches for the regex, an *ValueError* will be raised. Returns *None* if there isn't any match found.

#### **Parameters**

**key** – String key to query a layout.

# Returns

Corresponding layout based on the query.

**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.

```
\begin{aligned} &\textbf{popitem}() \to (k, \, v), \, \text{remove and return some (key, \, value) pair} \\ &\text{as a 2-tuple; but raise KeyError if D is empty.} \\ &\textbf{setdefault}(k\big[, \, d\,\big]) \to D.\text{get}(k, \! d), \, \text{also set D}[k] = d \, \text{if } k \, \text{not in D} \\ &\textbf{update}(\big[E, \,\big] **F) \to \text{None. Update D from mapping/iterable E and F.} \\ &\text{If E present and has a .keys() method, does: for k in E: D[k] = E[k] \, \text{If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v \, \text{In either case, this is followed by: for k, v in F.items(): D[k] = v} \\ &\textbf{values}() \to \text{an object providing a view on D's values} \end{aligned}
```

# AFL.automation.shared.PersistentConfig.PersistentConfig

Bases: MutableMapping

A dictionary-like class that serializes changes to disk

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

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

Constructor

# **Parameters**

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

#### **Methods**

init(path[, defaults, overrides, lock,])	Constructor	
clear()		

continues on next page

Table 129 - continued from previous page

<pre>get(key[, default])</pre>		Retrieves the corresponding layout by the string key.
<pre>get_historical_values(key[, vert_to_datetime])</pre>	con-	Convenience method for gathering historical values of a parameter
items()		of a parameter
1 CMS()		
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.
<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(key, default=None)
```

Retrieves the corresponding layout by the string key.

When there isn't an exact match, all the existing keys in the layout map will be treated as a regex and map against the input key again. When there are multiple matches for the regex, an *ValueError* will be raised. Returns *None* if there isn't any match found.

#### **Parameters**

**key** – String key to query a layout.

### **Returns**

Corresponding layout based on the query.

**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.

continues on next page

Table 130 – continued from previous page

	1 1 0
IPVersion(value)	
RunThread(coro)	
ServerDiscovery()	ServerDiscovery class
ServiceBrowser(zc, type_[, handlers,])	Used to browse for a service of a specific type.
ServiceInfo	Service information.
ServiceStateChange(value)	
Zeroconf([interfaces, unicast, ip_version,])	Implementation of Zeroconf Multicast DNS Service Discovery

# AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser

class AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser(zeroconf: Zeroconf, type\_: str |

list, handlers: ServiceListener |
List[Callable[[...], None]] |
None = None, listener:
ServiceListener | None = None,
addr: str | None = None, port:
int = 5353, delay: int = 10000,
question\_type:
DNSQuestionType | None =
None)

Bases: \_ServiceBrowserBase

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- *delay*: The initial delay between answering questions
- question\_type: The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

```
__init__(zeroconf: Zeroconf, type_: str | 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**

```
types

zc

query_scheduler

done

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.

types

zc

query\_scheduler

done

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.
	continues on next page

Table 133 - continued from previous page

async_update_records(zc, now, records)  async_update_records_complete()  async_update_records_complete()  Called when a record update has completed for all handlers.  Calling task waits for a given number of milliseconds or until notified.  dns_addresses([override_ttl, version])  dns_nsec(missing_types[, override_ttl])  Return matching DNSAddress from ServiceInfo.  dns_pointer([override_ttl])  Return DNSPointer from ServiceInfo.  dns_service([override_ttl])  Return DNSPointer from ServiceInfo.  dns_text([override_ttl])  Return DNSText from ServiceInfo.  Return DNSText from ServiceInfo.  Build a set of address records and NSEC records for non-present record types.  Name accessor  ip_addresses_by_version(version)  List ip_address objects matching IP version.  load_from_cache(zc[, now])  parsed_scoped_addresses([version])  parsed_scoped_addresses([version])  Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when available  request(zc, timeout[, question_type, addr, port])  Returns true if the service could be discovered on the network, and updates this object with details discovered.  set_server_if_missing()  Set the server if it is missing.  Update_record(zc, now, record)  Update a single record.</interface_index>		· · · · · · · · · · · · · · · · · · ·
async_wait(timeout[, loop])Calling task waits for a given number of milliseconds or until notified.dns_addresses([override_ttl, version])Return matching DNSAddress from ServiceInfo.dns_nsec(missing_types[, override_ttl])Return DNSNsec from ServiceInfo.dns_pointer([override_ttl])Return DNSPointer from ServiceInfo.dns_service([override_ttl])Return DNSService from ServiceInfo.dns_text([override_ttl])Return DNSText from ServiceInfo.get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>		Called when a record update has completed for all
or until notified.  dns_addresses([override_ttl, version]) Return matching DNSAddress from ServiceInfo.  dns_nsec(missing_types[, override_ttl]) Return DNSNsec from ServiceInfo.  dns_pointer([override_ttl]) Return DNSPointer from ServiceInfo.  dns_service([override_ttl]) Return DNSService from ServiceInfo.  dns_text([override_ttl]) Return DNSText from ServiceInfo.  get_address_and_nsec_records([override_ttl]) Build a set of address records and NSEC records for non-present record types.  get_name() Name accessor  ip_addresses_by_version(version) List ip_address objects matching IP version.  load_from_cache(zc[, now]) Populate the service info from the cache.  parsed_addresses([version]) Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when available  request(zc, timeout[, question_type, addr, port]) Returns true if the service could be discovered on the network, and updates this object with details discovered.  set_server_if_missing() Set the server if it is missing.</interface_index>		11411-01-01
dns_nsec(missing_types[, override_ttl])Return DNSNsec from ServiceInfo.dns_pointer([override_ttl])Return DNSPointer from ServiceInfo.dns_service([override_ttl])Return DNSService from ServiceInfo.dns_text([override_ttl])Return DNSText from ServiceInfo.get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>async_wait(timeout[, loop])</pre>	
dns_pointer([override_ttl])Return DNSPointer from ServiceInfo.dns_service([override_ttl])Return DNSService from ServiceInfo.dns_text([override_ttl])Return DNSText from ServiceInfo.get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>dns_addresses([override_ttl, version])</pre>	Return matching DNSAddress from ServiceInfo.
dns_pointer([override_ttl])Return DNSPointer from ServiceInfo.dns_service([override_ttl])Return DNSService from ServiceInfo.dns_text([override_ttl])Return DNSText from ServiceInfo.get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>dns_nsec(missing_types[, override_ttl])</pre>	Return DNSNsec from ServiceInfo.
dns_service([override_ttl])Return DNSService from ServiceInfo.dns_text([override_ttl])Return DNSText from ServiceInfo.get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>dns_pointer([override_ttl])</pre>	Return DNSPointer from ServiceInfo.
get_address_and_nsec_records([override_ttl])Build a set of address records and NSEC records for non-present record types.get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>		Return DNSService from ServiceInfo.
non-present record types.  get_name()	<pre>dns_text([override_ttl])</pre>	Return DNSText from ServiceInfo.
get_name()Name accessorip_addresses_by_version(version)List ip_address objects matching IP version.load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>get_address_and_nsec_records([override_ttl])</pre>	
load_from_cache(zc[, now])Populate the service info from the cache.parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>get_name()</pre>	
parsed_addresses([version])List addresses in their parsed string form.parsed_scoped_addresses([version])Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when availablerequest(zc, timeout[, question_type, addr, port])Returns true if the service could be discovered on the network, and updates this object with details discovered.set_server_if_missing()Set the server if it is missing.</interface_index>	<pre>ip_addresses_by_version(version)</pre>	List ip_address objects matching IP version.
parsed_scoped_addresses([version])       Equivalent to parsed_addresses, with the exception that IPv6 Link-Local addresses are qualified with % <interface_index> when available         request(zc, timeout[, question_type, addr, port])       Returns true if the service could be discovered on the network, and updates this object with details discovered.         set_server_if_missing()       Set the server if it is missing.</interface_index>	<pre>load_from_cache(zc[, now])</pre>	Populate the service info from the cache.
that IPv6 Link-Local addresses are qualified with % <interface_index> when available  request(zc, timeout[, question_type, addr, port])  Returns true if the service could be discovered on the network, and updates this object with details discovered.  set_server_if_missing()  Set the server if it is missing.</interface_index>	parsed_addresses([version])	List addresses in their parsed string form.
network, and updates this object with details discovered.  set_server_if_missing()  Set the server if it is missing.	<pre>parsed_scoped_addresses([version])</pre>	that IPv6 Link-Local addresses are qualified with
- "	<pre>request(zc, timeout[, question_type, addr, port])</pre>	network, and updates this object with details discov-
<pre>update_record(zc, now, record)</pre> Update a single record.	<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**

text	
type	
key	
port	
weight	
priority	
server	
server_key	
host_ttl	
other_ttl	
<pre>interface_index</pre>	
addresses	IPv4 addresses of this service.
decoded_properties	Return properties as strings.

continues on next page

Table 134 - continued from previous page

name	The name of the service.
properties	Return properties as bytes.

text

type

key

port

weight

priority

server

server\_key

host\_ttl

other\_ttl

### interface\_index

```
__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 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 \mid 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.

 $dns_pointer(override_ttl: int \mid None = None) \rightarrow DNSPointer$ 

Return DNSPointer from ServiceInfo.

 $dns_service(override\_ttl: int \mid None = None) \rightarrow DNSService$ 

Return DNSService from ServiceInfo.

```
dns_text(override_ttl: int \mid 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.

```
\texttt{get\_name()} \to 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.

```
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.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], 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], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False, zc: Zeroconf | None = None) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, and listening.

### **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

```
async async_register_service(info: ServiceInfo, ttl: int | None = None, allow_name_change: bool = False, cooperating_responders: bool = False, strict: bool = True) \rightarrow Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service. The name of the service may be changed if needed to make it unique on the network. Additionally multiple cooperating responders can register the same service on the network for resilience (if you want this behavior set *cooperating\_responders* to *True*).

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

# async async\_unregister\_all\_services() $\rightarrow$ None

Unregister all registered services.

Unlike async\_register\_service and async\_unregister\_service, this method does not return a future and is always expected to be awaited since its only called at shutdown.

```
async async_unregister_service(info: ServiceInfo) → Awaitable
```

Unregister a service.

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

```
async async_update_service(info: ServiceInfo) → Awaitable
```

Registers service information to the network with a default TTL. Zeroconf will then respond to requests for information for that service.

The service will be broadcast in a task. This task is returned and therefore can be awaited if necessary.

```
async async_close() \rightarrow None
```

Ends the background threads, and prevent this instance from servicing further queries.

```
async async_get_service_info(type_: str, name: str, timeout: int = 3000, question_type: DNSQuestionType \mid None = None) \rightarrow AsyncServiceInfo \mid None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

# **Parameters**

- type fully qualified service type name
- name the name of the service
- timeout milliseconds to wait for a response
- question\_type The type of questions to ask (DNSQuestionType.QM or DNSQuestionType.QU)

async\_add\_service\_listener( $type_{-}$ : str, listener: ServiceListener)  $\rightarrow$  None

Adds a listener for a particular service type. This object will then have its add\_service and remove\_service methods called when services of that type become available and unavailable.

async async\_remove\_service\_listener( $listener: ServiceListener) \rightarrow None$ 

Removes a listener from the set that is currently listening.

async async\_remove\_all\_service\_listeners()  $\rightarrow$  None

Removes a listener from the set that is currently listening.

# AFL.automation.shared.ServerDiscovery.AsyncZeroconfServiceTypes

# class AFL.automation.shared.ServerDiscovery.AsyncZeroconfServiceTypes

Bases: ZeroconfServiceTypes

An async version of ZeroconfServiceTypes.

**\_\_init\_\_()**  $\rightarrow$  None

Keep track of found services in a set.

#### **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:  $Sequence[str | int | Tuple[Tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, <math>ip\_version$ : IPVersion | None = None)  $\rightarrow$  Tuple[str, ...]

Return all of the advertised services on any local networks.

# **Parameters**

- aiozc AsyncZeroconf() instance. Pass in if already have an instance running or if nondefault interfaces are needed
- timeout seconds to wait for any responses
- interfaces interfaces to listen on.
- **ip\_version** IP protocol version to use.

#### Returns

tuple of service type strings

**\_\_init\_\_()**  $\rightarrow$  None

Service added.

Keep track of found services in a set.

 $\mathbf{add\_service}(\mathit{zc} \colon \mathit{Zeroconf}, \mathit{type}\_: \mathit{str}, \mathit{name} \colon \mathit{str}) \to \mathit{None}$ 

```
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) → None
    Service removed.
update_service(zc: Zeroconf, type_: str, name: str) → None
    Service updated.
```

# AFL.automation.shared.ServerDiscovery.IPVersion

```
class AFL.automation.shared.ServerDiscovery.IPVersion(value)
```

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

## **Attributes**

```
V40n1y
V60n1y
A11
```

```
V4Only = 1
V6Only = 2
All = 3
```

classmethod \_\_contains\_\_(member)

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching name.

```
classmethod __iter__()
```

Return members in definition order.

# classmethod \_\_len\_\_()

Return the number of members (no aliases)

# AFL.automation.shared.ServerDiscovery.RunThread

# **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>	
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>	

continues on next page

Table 140 – continued from previous page

_		
	<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	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.

```
__init__()
```

**on\_service\_state\_change**(*zeroconf*: Zeroconf, *service\_type*: *str*, *name*: *str*, *state\_change*: ServiceStateChange) → 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.ServerDiscovery.ServiceBrowser

None | | | None = None, listener: ServiceListener | None = None, addr: str | None = None, port: int = 5353, delay: int = 10000, question\_type: DNSQuestionType | None = None)

Bases: \_ServiceBrowserBase, Thread

Used to browse for a service of a specific type.

The listener object will have its add\_service() and remove\_service() methods called when this browser discovers changes in the services availability.

```
__init__(zc: Zeroconf, type_: str | list, handlers: ServiceListener | List[Callable[..., None]] | None = None, listener: ServiceListener | None = None, addr: str | None = None, port: int = 5353, delay: int = 10000, question_type: DNSQuestionType | None = None) → None
```

Used to browse for a service for specific type(s).

Constructor parameters are as follows:

- zc: A Zeroconf instance
- type\_: fully qualified service type name
- handler: ServiceListener or Callable that knows how to process ServiceStateChange events
- listener: ServiceListener
- addr: address to send queries (will default to multicast)
- port: port to send queries (will default to mdns 5353)
- delay: The initial delay between answering questions
- 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**

types	
zc	
query_scheduler	
done	
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	

# types

zc

## query\_scheduler

### done

\_\_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

# ${\bf class} \ {\tt AFL.} automation. shared. Server {\tt Discovery.} {\bf Service Info}$

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**

text			
type			

continues on next page

Table 144 – continued from previous page

key port  weight  priority  server  server_key  host_ttl  other_ttl  interface_index  addresses IPv4 addresses of this service. decoded_properties Return properties as strings. name The name of the service. properties Return properties as bytes.		
weight  priority  server  server_key  host_ttl  other_ttl  interface_index  addresses  decoded_properties name  The name of the service.	key	
priority  server  server_key  host_ttl  other_ttl  interface_index  addresses  decoded_properties name  The name of the service.	port	
server_key  host_ttl  other_ttl  interface_index  addresses	weight	
server_key  host_ttl  other_ttl  interface_index  addresses  decoded_properties  Return properties as strings.  name  The name of the service.	priority	
host_ttl  other_ttl  interface_index  addresses  decoded_properties  name  IPv4 addresses of this service.  Return properties as strings.  The name of the service.	server	
<pre>interface_index  addresses</pre>	server_key	
<pre>interface_index  addresses</pre>	host_ttl	
addressesIPv4 addresses of this service.decoded_propertiesReturn properties as strings.nameThe name of the service.	other_ttl	
decoded_propertiesReturn properties as strings.nameThe name of the service.	interface_index	
name The name of the service.	addresses	IPv4 addresses of this service.
name The name of the service.	decoded_properties	Return properties as strings.
properties Return properties as bytes.		
	properties	Return properties as bytes.

text

type

key

port

weight

priority

server

server\_key

host\_ttl

other\_ttl

interface\_index

\_\_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 async_wait(timeout: float, loop: AbstractEventLoop | None = None) \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 \mid None = None, version: IPVersion = IPVersion.All) \rightarrow List[DNSAddress] Return matching DNSAddress from ServiceInfo.
```

```
\textbf{dns\_nsec}(\textit{missing\_types: List[int]}, \textit{override\_ttl: int} \mid \textit{None} = \textit{None}) \rightarrow \text{DNSNsec}
```

Return DNSNsec from ServiceInfo.

```
dns_pointer(override_ttl: int | None = None) \rightarrow DNSPointer
```

Return DNSPointer from ServiceInfo.

```
dns_service(override_ttl: int \mid 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

```
class AFL.automation.shared.ServerDiscovery.ServiceStateChange(value)
```

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

#### **Attributes**

```
Added
Removed
Updated
```

```
Added = 1
```

Removed = 2

Updated = 3

```
classmethod __contains__(member)
```

Return True if member is a member of this enum raises TypeError if member is not an enum member note: in 3.12 TypeError will no longer be raised, and True will also be returned if member is the value of a member in this enum

```
classmethod __getitem__(name)
```

Return the member matching name.

```
classmethod __iter__()
```

Return members in definition order.

#### classmethod \_\_len\_\_()

Return the number of members (no aliases)

## AFL.automation.shared.ServerDiscovery.Zeroconf

```
class AFL.automation.shared.ServerDiscovery.Zeroconf(interfaces: Sequence[str | int | Tuple[Tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip\_version: IPVersion | None = None, apple\_p2p: bool = False)
```

Bases: QuietLogger

Implementation of Zeroconf Multicast DNS Service Discovery

Supports registration, unregistration, queries and browsing.

```
__init__(interfaces: Sequence[str | int | Tuple[Tuple[str, int, int], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False, ip_version: IPVersion | None = None, apple_p2p: bool = False) \rightarrow None
```

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

#### **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for Interface-Choice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

#### **Methods**

init([interfaces, unicast, ip_version,])	Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.
<pre>add_listener(listener, question)</pre>	Adds a listener for a given question.
<pre>add_service_listener(type_, listener)</pre>	Adds a listener for a particular service type.
<pre>async_add_listener(listener, question)</pre>	Adds a listener for a given question.
<pre>async_check_service(info, allow_name_change)</pre>	Checks the network for a unique service name, modifying the ServiceInfo passed in if it is not unique.
<pre>async_get_service_info(type_, name[,])</pre>	Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.
async_notify_all()	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.
<pre>async_unregister_service(info)</pre>	Unregister a service.
async_update_service(info)	Registers service information to the network with a default TTL.
<pre>async_wait(timeout)</pre>	Calling task waits for a given number of milliseconds or until notified.
async_wait_for_start()	Wait for start up for actions that require a running Zeroconf instance.

continues on next page

Table 146 – continued from previous page

close()	Ends the background threads, and prevent this in-
	stance 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.
unregister_all_services()	Unregister all registered services.
unregister_service(info)	Unregister a service.
<pre>update_service(info)</pre>	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], int]] | InterfaceChoice = InterfaceChoice.All, unicast: bool = False,  $ip\_version$ : IPVersion | None = None,  $apple\_p2p$ : bool = False)  $\rightarrow$  None

Creates an instance of the Zeroconf class, establishing multicast communications, listening and reaping threads.

## **Parameters**

• **interfaces** – InterfaceChoice or a list of IP addresses (IPv4 and IPv6) and interface indexes (IPv6 only).

IPv6 notes for non-POSIX systems: \* InterfaceChoice.All is an alias for InterfaceChoice.Default

on Python versions before 3.8.

Also listening on loopback (::1) doesn't work, use a real address.

- **ip\_version** IP versions to support. If *choice* is a list, the default is detected from it. Otherwise defaults to V4 only for backward compatibility.
- apple\_p2p use AWDL interface (only macOS)

#### property started: bool

Check if the instance has started.

```
start() \rightarrow None
```

Start Zeroconf.

```
async async_wait_for_start() \rightarrow None
```

Wait for start up for actions that require a running Zeroconf instance.

Throws NotRunningException if the instance is not running or could not be started.

```
property listeners: Set[RecordUpdateListener]
```

```
async async_wait(timeout: float) \rightarrow None
```

Calling task waits for a given number of milliseconds or until notified.

```
notify_all() \rightarrow None
```

Notifies all waiting threads and notify listeners.

```
async\_notify\_all() \rightarrow None
```

Schedule an async\_notify\_all.

```
\texttt{get\_service\_info}(type\_: str, name: str, timeout: int = 3000, question\_type: DNSQuestionType | None = None) <math>\rightarrow ServiceInfo | None
```

Returns network's service information for a particular name and type, or None if no service matches by the timeout, which defaults to 3 seconds.

## **Parameters**

- type fully qualified service type name
- name the name of the service
- timeout milliseconds to wait for a response
- question\_type The type of questions to ask (DNSQuestionType.QM or 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 *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 \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)

```
\label{eq:generate_service_broadcast} \textit{(info: ServiceInfo, ttl: int | None, broadcast\_addresses: bool = True)} \rightarrow DNSOutgoing
```

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.

 $\textbf{add\_listener}(\textit{listener}: \textit{RecordUpdateListener}, \textit{question}: \textit{DNSQuestion} \mid \textit{List[DNSQuestion]} \mid \textit{None}) \rightarrow \\ \text{None}$ 

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 \mid None = None, port: int = 5353, v6\_flow\_scope: Tuple \mid Tuple[int, int] = (), transport: \_WrappedTransport \mid 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 | Tuple[int, int] = (), transport: \_WrappedTransport \mid None = None) \rightarrow None
```

Sends an outgoing packet.

```
\textbf{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 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

## AFL.automation.shared.exceptions.EmptyException

exception AFL.automation.shared.exceptions.EmptyException

Raised when a mixture runs out of mass or volume

## AFL.automation.shared.exceptions.MixingException

exception AFL.automation.shared.exceptions.MixingException

Raised when a mixture cannot be made

## AFL.automation.shared.exceptions.NoDeviceFoundException

exception AFL.automation.shared.exceptions.NoDeviceFoundException

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

## AFL.automation.shared.exceptions.NotFoundError

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.widgetui

#### **Functions**

clear_output([wait])	Clear the output of the current cell receiving output.
<pre>client_construct_ui(self[, return_ui,])</pre>	
1' 7 (% 1 · Γ · 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	D: 1 D d 1: (: 116 ( 1
<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
      }
}
```

```
>>> 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
```



update\_display()

#### **Notes**

In Python, objects can declare their textual representation using the <u>\_\_repr\_\_</u> method. IPython expands on this idea and allows objects to declare other, rich representations including:

- HTML
- JSON
- PNG
- JPEG
- SVG
- LaTeX

A single object can declare some or all of these representations; all are handled by IPython's display system.

The main idea of the first approach is that you have to implement special display methods when you define your class, one for each representation you want to use. Here is a list of the names of the special methods and the values they must return:

- \_repr\_html\_: return raw HTML as a string, or a tuple (see below).
- \_repr\_json\_: return a JSONable dict, or a tuple (see below).
- \_repr\_jpeg\_: return raw JPEG data, or a tuple (see below).
- \_repr\_png\_: return raw PNG data, or a tuple (see below).
- \_repr\_svg\_: return raw SVG data as a string, or a tuple (see below).
- \_repr\_latex\_: return LaTeX commands in a string surrounded by "\$", or a tuple (see below).
- \_repr\_mimebundle\_: return a full mimebundle containing the mapping from all mimetypes to data. Use this for any mime-type not listed above.

The above functions may also return the object's metadata alonside the data. If the metadata is available, the functions will return a tuple containing the data and metadata, in that order. If there is no metadata available, then the functions will return the data only.

When you are directly writing your own classes, you can adapt them for display in IPython by following the above approach. But in practice, you often need to work with existing classes that you can't easily modify.

You can refer to the documentation on integrating with the display system in order to register custom formatters for already existing types (integrating\_rich\_display).

Added in version 5.4: display available without import

Added in version 6.1: display available without import

Since IPython 5.4 and 6.1 *display()* is automatically made available to the user without import. If you are using display in a document that might be used in a pure python context or with older version of IPython, use the following import at the top of your file:

from IPython.display import display

#### **Classes**

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

## AFL.automation.shared.widgetui.Markdown

Bases: TextDisplayObject

\_\_init\_\_(data=None, url=None, filename=None, metadata=None)

Create a display object given raw data.

When this object is returned by an expression or passed to the display function, it will result in the data being displayed in the frontend. The MIME type of the data should match the subclasses used, so the Png subclass should be used for 'image/png' data. If the data is a URL, the data will first be downloaded and then displayed.

#### **Parameters**

- data (unicode, str or bytes) The raw data or a URL or file to load the data from
- **url** (*unicode*) A URL to download the data from.
- **filename** (*unicode*) Path to a local file to load the data from.
- metadata (dict) Dict of metadata associated to be the object when displayed

## **Methods**

init([data, url, filename, metadata])	Create a display object given raw data.
reload()	Reload the raw data from file or URL.

## **Attributes**

metadata

**\_\_init\_\_**(data=None, url=None, filename=None, metadata=None)

Create a display object given raw data.

When this object is returned by an expression or passed to the display function, it will result in the data being displayed in the frontend. The MIME type of the data should match the subclasses used, so the Png subclass should be used for 'image/png' data. If the data is a URL, the data will first be downloaded and then displayed.

#### **Parameters**

- data (unicode, str or bytes) The raw data or a URL or file to load the data from
- url (unicode) A URL to download the data from.
- **filename** (*unicode*) Path to a local file to load the data from.
- metadata (dict) Dict of metadata associated to be the object when displayed

metadata = None

reload()

Reload the raw data from file or URL.

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

**CHAPTER** 

**TWO** 

## **API REFERENCE**

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

AFL.automation. APIServer

AFL.automation.instrument

AFL.automation.loading

AFL.automation.sample

AFL.automation.sample

## 2.1 AFL.automation

## **Functions**

test() Run all tests using pytest.

AFL.automation.test()
Run all tests using pytest.

## **Modules**

APIServer

EpicsADLiveProcess

instrument

loading

continues on next page

## Table 3 – continued from previous page

```
sample_env
shared
```

## 2.1.1 AFL.automation.APIServer

#### **Modules**

```
Client
LoggerFilter
QueueDaemon
```

#### 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.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 for testing only that takes all its data and simply throws it away.

QueueDaemon(app, driver, task\_queue, history)

**\_\_init\_\_**(app, driver, task\_queue, history, debug=False, data=None)

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

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

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

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

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

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

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

## terminate()

check\_if\_paused()

mask\_serialized\_objs(package)

run()

Method representing the thread's activity.

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

## 2.1.2 AFL.automation.EpicsADLiveProcess

#### **Modules**

Client

ReduceDaemon

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

```
class AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon(app, reduction\_queue, integrator, results, mask=None, npts=500)
```

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

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

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

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

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

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

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

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

#### terminate()

run()

Method representing the thread's activity.

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

## 2.1.3 AFL.automation.instrument

#### **Modules**

FileCamera

NetworkCamera

#### AFL.automation.instrument.FileCamera

#### **Classes**

```
FileCamera()
```

class AFL.automation.instrument.FileCamera.FileCamera

```
__init__()
collect(fname)
```

## AFL.automation.instrument.NetworkCamera

## **Classes**

```
NetworkCamera(url)
```

class AFL.automation.instrument.NetworkCamera.NetworkCamera(url)

```
__init__(url)
```

## collect()

## 2.1.4 AFL.automation.loading

## **Modules**

CetoniMultiPosValve	
ChemyxSyringePump	This is largely duplicated from the reference code provided by Chemyx.
DigitalOutPressureController	
DoubleViciMultiposSelector	
DummyPump	
FlowSelector	
MultiChannelRelay	
NE1kSyringePump	
PressureController	
PressureControllerAsPump	
SainSmartRelay	
SampleCell	
Sensor	
SensorCallbackThread	
SensorPollingThread	
SerialDevice	
SyringePump	
Tubing	
UltimusVPressureController	
ViciMultiposSelector	

## $AFL. automation. Ioading. Cetoni {\color{blue}MultiPosValve}$

## 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.
class AFL.automation.loading.ChemyxSyringePump.ChemyxConnection(port, baudrate, x=0, mode=0,
   verbose=False)
     __init__(port, baudrate, x=0, mode=0, verbose=False)
     openConnection()
     closeConnection()
     sendCommand(command)
```

```
getResponse()
startPump()
stopPump()
pausePump()
restartPump()
setUnits(units)
setDiameter(diameter)
setRate(rate)
setVolume(volume)
setDelay(delay)
setTime(timer)
getParameterLimits()
getParameters()
getDisplacedVolume()
getElapsedTime()
getPumpStatus()
addMode(command)
addX(command)
```

## AFL.automation.loading.DigitalOutPressureController

#### **Classes**

```
DigitalOutPressureController(digital_out, ...)

PressureController()

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

 $\textbf{class} \ \, \textbf{AFL}. \textbf{automation.loading.DigitalOutPressureController.DigitalOutPressureController} ( \textit{digital\_out}, \\ \textit{pres-sure\_to\_v\_conv})$ 

```
__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()

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

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,'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()
```

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

#### **Classes**

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

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

Abstract superclass for pressure controllers that provides timed dispensing

timed\_dispense(dispense\_pressure, dispense\_time, block=True)

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

### blockUntilStatusStopped(pollingdelay=0.2)

block execution until the controller finishes a dispense

## dispenseRunning()

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

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

#### stop()

Abort the current timed dispense action.

## AFL.automation.loading.PressureControllerAsPump

#### **Classes**

```
PressureControllerAsPump(pressure_controller)

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

SyringePump()
```

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

## 

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':FE050009000F4\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':FE05000DF00F0\r\n', b':FE05000DFF00F1\r\n'], [b':FE05000EFF00F0\r\n', b':FE05000EFF00F0\r\n'], [b':FE05000FF00FF\r\n'], b':FE05000EFF00F0\r\n'], [b':FE05000FF00FF\r\n'], b':FE05000EFF00FF\r\n'], b':FE05000FF00FF\r\n'], b':FE05000FFF00FF\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 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...

 $status = usbrelay[0][1] \# row-0 (status) stat\_ret = usbrelay[0][0] \# row-0 (status return) ch\_1\_on = usbrelay[1][1] \# row-1 (chan-1 off) ch\_1\_off = usbrelay[1][0] \# row-1 (chan-1 off) ch\_16\_on = usbrelay[16][1] \# row-16 (chan-16 on) ch\_16\_off = usbrelay[16][0] \# row-16 (chan-16 off) all\_on = usbrelay[17][1] \# row-17 (all on) all\_off = usbrelay[17][0] \# row-17 (all off)$ 

٠,,,,

## AFL.automation.loading.SampleCell

#### **Classes**

```
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.

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.

```
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()

 $\textbf{class} \ AFL. automation. loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread. \textbf{StopLoadCBv2} (poll, period, load\_client=None, load) and loading. Sensor Callback Thread thr$ 

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() load\_object=None)

**class** AFL.automation.loading.SensorCallbackThread.**LoaderCommunication**(load\_client=None,

```
__init__(load client=None, load object=None)
getServerState()
stopLoad()
```

## AFL.automation.loading.SensorPollingThread

#### Classes

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

\_\_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

withdraw(volume, block=True)

dispense(volume, block=True)

# Classes

```
SyringePump()

class AFL.automation.loading.SyringePump.SyringePump

stop()
```

# setRate(rate) getRate(rate)

emptySyringe()

# AFL.automation.loading.Tubing

#### Classes

```
Tubing(specid, length)
```

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

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

## AFL.automation.loading.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, ...])
class AFL.automation.loading.ViciMultiposSelector.ViciMultiposSelector(port, baudrate=9600,
  portlabels=None)
     __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,'instru-
                    ment':4,'rinse':5,'waste':6}
                  • {'sample' (of the form) - 3,'instrument':4,'rinse':5,'waste':6}
```

2.1. AFL.automation 179

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

# 2.1.5 AFL.automation.sample\_env

#### **Modules**

# 2.1.6 AFL.automation.shared

## **Modules**

DataLabelerWidget
DatasetWidget
DiffractionLabeler
MutableQueue
PersistentConfig
ServerDiscovery
exceptions
serialization
widgetui

# AFL.automation.shared.DataLabelerWidget

# **Functions**

sqrt(x,/)	Return the square root of x.
24= 2(11,7)	Treatment square root of m

## **Classes**

```
DataLabelerModel(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)
```

# AFL.automation.shared.DatasetWidget

## **Classes**

```
 \textbf{class} \  \, \textbf{AFL}. \, \textbf{automation}. \, \textbf{shared}. \, \textbf{DatasetWidget}. \, \textbf{DatasetWidget}( \, \textit{dataset: Dataset, sample\_dim: str} = \\ \, \textit{'sample', scatt\_variables: List[str]} \, | \, \textit{None} = \\ \, \textit{None, comps\_variable: str} \, | \, \textit{None} = \textit{None, comps\_color\_variable: str} \, | \, \textit{None} = \textit{None, comps\_color\_variable: str} \, | \, \textit{None} = \textit{None, xmin: float} = 0.001, \, \textit{xmax: float} = 1.0)
```

```
__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.rum()
   ``` next\_button\_callback(\*args)
  prev\_button\_callback(\*args)
  goto\_callback(\*args)
  composition\_click\_callback(figure, location, click)
  update\_composition\_plot()
  update\_scattering\_plot()

```
reset_dataset(*args)
update_dropdowns(*args)
update_sample_dim(*args)
```

update\_plots()

initialize\_plots(\*args)

update\_colors(\*args)

apply\_sel(\*args)

apply\_isel(\*args)

extract\_var(\*args)

combine\_vars(\*args)

get\_comps()

```
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] |
                                                                      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.
```

## **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
	•

# **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

#### **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.
IPVersion(value)	
RunThread(coro)	
ServerDiscovery()	ServerDiscovery class
ServiceBrowser(zc, type_[, handlers,])	Used to browse for a service of a specific type.
ServiceInfo	Service information.
ServiceStateChange(value)	
Zeroconf([interfaces, unicast, ip_version,])	Implementation of Zeroconf Multicast DNS Service Discovery

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

## \_\_init\_\_(coro)

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

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

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

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

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

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

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

## run()

Method representing the thread's activity.

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

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

ServerDiscovery class

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

```
__init__()
```

on\_service\_state\_change(zeroconf: Zeroconf, service\_type: str, name: str, state\_change:
ServiceStateChange) → 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
	continues on next page

Table	50	<ul><li>continued</li></ul>	from	previous r	bage

NoDeviceFoundException	Raised when no matching device can be found on the selected port
NotFoundError	
SerialCommsException	Raised when the system receives a serial response it can't parse, likely a garbled line

## exception AFL.automation.shared.exceptions.EmptyException

Raised when a mixture runs out of mass or volume

# exception AFL.automation.shared.exceptions.MixingException

Raised when a mixture cannot be made

## 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)

# **Exceptions**

 ${\it UnpicklingError}$ 

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

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

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

# 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>	

continues on next page

# Table 53 – continued from previous page

display(\*objs[, include, exclude, metadata, ...])

Display a Python object in all frontends.

# **Classes**

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

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

# 2.2 AFL.automation.sample

# **Modules**

\_\_\_

**CHAPTER** 

# **THREE**

# MODULE DOCUMENTATION

AFL.automation

AFL.automation.APIServer

AFL.automation.instrument

AFL.automation.loading

AFL.automation.sample

AFL.automation.sample\_env

AFL.automation.shared

# 3.1 AFL.automation

# **Functions**

test() Run all tests using pytest.

# 3.1.1 AFL.automation.test

AFL.automation.test()

Run all tests using pytest.

AFL.automation.test()

Run all tests using pytest.

# Modules

*APIServer* 

*EpicsADLiveProcess* 

continues on next page

# Table 3 – continued from previous page

instrument
loading
sample\_env
shared

# 3.1.2 AFL.automation.EpicsADLiveProcess

#### **Modules**

Client

ReduceDaemon

# 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'*)

# **Methods**

\_\_*init*\_\_([ip, port])

halt()

home()

load\_instrument(name, mount, tip\_rack\_slots)

load\_labware(name, slot)

logged\_in()

continues on next page

# Table 6 – continued from previous page

```
login(username)
       reset()
       set_queue_mode([debug_mode])
                                                          Transfer fluid from one location to another
       transfer(source, dest, volume[, source_loc, ...])
     __init__(ip='10.42.0.30', port='5000')
     logged_in()
     login(username)
     set_queue_mode(debug_mode=True)
     transfer(source, dest, volume, source_loc=None, dest_loc=None)
           Transfer fluid from one location to another
               Parameters
                   • source (str or list of str) - Source wells to transfer from. Wells should be spec-
                     ified as three character strings with the first character being the slot number.
                   • dest (str or list of str) - Destination wells to transfer from. Wells should be spec-
                     ified as three character strings with the first character being the slot number.
                   • volume (float) – volume of fluid to transfer in microliters
     load_labware(name, slot)
     load_instrument(name, mount, tip_rack_slots)
     reset()
     home()
     halt()
class AFL.automation.EpicsADLiveProcess.Client.Client(ip='10.42.0.30', port='5000')
     Communicate with NistoRoboto server on OT-2
     This class maps pipettor functions to HTTP REST requests that are sent to the NistoRoboto server
     __init__(ip='10.42.0.30', port='5000')
     logged_in()
     login(username)
     set_queue_mode(debug_mode=True)
     transfer(source, dest, volume, source_loc=None, dest_loc=None)
           Transfer fluid from one location to another
               Parameters
```

• **source** (*str or list of str*) – Source wells to transfer from. Wells should be specified as three character strings with the first character being the slot number.

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

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

# AFL.automation.EpicsADLiveProcess.ReduceDaemon

#### **Classes**

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

# AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon

Bases: Thread

**\_\_init\_\_**(app, reduction\_queue, integrator, results, mask=None, npts=500)

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

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

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

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

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

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

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

# **Methods**

init(app, reduction_queue, integrator,)	This constructor should always be called with keyword arguments.
<pre>getName()</pre>	Return a string used for identification purposes only.
isDaemon()	Return whether this thread is a daemon.
is_alive()	Return whether the thread is alive.
<pre>join([timeout])</pre>	Wait until the thread terminates.
run()	Method representing the thread's activity.
setDaemon(daemonic)	Set whether this thread is a daemon.

continues on next page

Table 8 – continued from previous page

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.

#### 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

# **FOUR**

# **INDICES AND TABLES**

- genindex
- modindex
- search

# **PYTHON MODULE INDEX**

```
а
                                                        179
                                               AFL.automation.sample, 191
AFL. automation, 157
                                               AFL.automation.sample_env, 180
AFL.automation.APIServer, 158
                                               AFL.automation.shared, 180
AFL.automation.APIServer.Client, 158
                                               AFL.automation.shared.DataLabelerWidget, 180
AFL.automation.APIServer.LoggerFilter, 159
                                               AFL.automation.shared.DatasetWidget, 182
AFL.automation.APIServer.QueueDaemon, 160
                                               AFL.automation.shared.DiffractionLabeler, 185
AFL.automation.EpicsADLiveProcess, 160
                                               AFL.automation.shared.exceptions, 189
AFL.automation.EpicsADLiveProcess.Client, 161
{\tt AFL.automation.EpicsADLiveProcess.ReduceDaemor}, {\tt FL.automation.shared.MutableQueue}, 186 \\
                                                AFL.automation.shared.PersistentConfig, 186
                                               AFL.automation.shared.serialization, 190
AFL.automation.instrument. 162
                                               AFL.automation.shared.ServerDiscovery, 188
AFL.automation.instrument.FileCamera, 162
                                               AFL.automation.shared.widgetui, 190
AFL.automation.instrument.NetworkCamera, 162
AFL.automation.loading, 163
AFL.automation.loading.CetoniMultiPosValve,
AFL.automation.loading.ChemyxSyringePump, 164
AFL.automation.loading.DigitalOutPressureController,
AFL.automation.loading.DoubleViciMultiposSelector,
        167
AFL.automation.loading.DummyPump, 167
AFL.automation.loading.FlowSelector, 168
AFL.automation.loading.MultiChannelRelay, 168
AFL.automation.loading.NE1kSyringePump, 168
AFL.automation.loading.PressureController,
AFL.automation.loading.PressureControllerAsPump,
        170
AFL.automation.loading.SainSmartRelay, 171
AFL.automation.loading.SampleCell, 172
AFL.automation.loading.Sensor, 172
AFL.automation.loading.SensorCallbackThread,
AFL.automation.loading.SensorPollingThread,
        176
AFL.automation.loading.SerialDevice, 177
AFL.automation.loading.SyringePump, 178
AFL.automation.loading.Tubing, 178
AFL.automation.loading.UltimusVPressureController,
```

AFL.automation.loading.ViciMultiposSelector,

204 Python Module Index

# **INDEX**

Symbols	method), 27
contains() (AFL.automation.shared.ServerDiscovery.Hingit	$_{on}$ () (AFL.automation.loading.DoubleViciMultiposSelector.Double
class method) 130	method), 28, 29, 31, 107
contains() (AFL.automation.shared.ServerDiscovery.Services	method), 29
getitem() (AFL.automation.shared.PersistentConfig.Persistent	method), 30
getitem() (AFL.automation.shared.ServerDiscovery.HPVerston	() (AFL.automation.loading.DoubleViciMultiposSelector.ViciMu method), 30
class method), 130getitem() (AFL.automation.shared.ServerDiscovery.Service	tale (AFI) automation.loading.DummyPump.DummyPump
class method), 143	method), 32, 34, 167 _() (AFL.automation.loading.DummyPump.SerialDevice
mathod) 2 5 8 158	method), 33
init() (AFL.automation.APIServer.Client.ServerDiscoveryit_	_() (AFL.automation.loading.DummyPump.SyringePump method), 33
method), 6, 7init() (AFL.automation.APIServer.LoggerFilter.LoggerFilter-	_() (AFL.automation.loading.FlowSelector.FlowSelector
method) 0 150	memou), 54
init() (AFL.automation.APIServer.QueueDaemon.Datainish	က်(AFL.automation.loading.MultiChannelRelay.MultiChannelR method), 35
method), 10, 11	
init() (AFL.automation.APIServer.QueueDaemon.QueileDaemon.queileDa	meinoa), 30, 37, 39, 109
init() (AFL.automation.EpicsADLiveProcess.Client.Client	_() (AFL.automation.loading.NE1kSyringePump.SerialDevice method), 38
method), 161, 194, 195	(AFI automation loading NF1kSvringePumn SvringePumn
init() (AFL.automation.EpicsADLiveProcess.Reduce <del>Ddemon.</del> method), 161, 196, 197, 199	meinoa), 38
init() (AFL.automation.instrument.FileCamera.FileCamera	() (AFL.automation.loading.PressureController.PressureContro
method), 16, 162	method), 40  (AFI automation loading PressureController As Pump Pressure
init() (AFL.automation.instrument.NetworkCamera.Networ	metnoa), 41-43, 170
init() (AFL.automation.loading.CetoniMultiPosValve.CetoniM	meinoa), 42, 43
init() (AFL.automation.loading.CetoniMultiPosValve.Fibitse	$_{le}(Q)$ AFL.automation.loading.PressureControllerAsPump.Syringe
method), 19	method), 43
init() (AFL.automation.loading.ChemyxSyringePump.Chemys method), 20, 21, 25, 165	x&dihAetiantomation.loading.SainSmartRelay.MultiChannelRelay method), 45
init() (AFL.automation.loading.ChemyxSyringePump.Chemyx	(AFF) automation.loading.SainSmartRelay.SainSmartRelay method), 45–47, 171
method), 22–24, 165init() (AFL.automation.loading.ChemyxSyringePump.Syringe	
method), 24init() (AFL.automation.loading.DigitalOutPressureControlle	r. Dighalibantopsation loading Sample Cell. Sample Cell
method), 26, 28, 166	method), 48
init() (AFL.automation.loading.DigitalOutPressureControlle	r.Presstire cummitation.10aaing.sensor.Dummysensor1

```
method), 49, 54, 173
                                                                                               method), 105, 112, 185
 __init__() (AFL.automation.loading.Sensor.DummySenso<u>r2</u>init__() (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
             method), 51, 52, 55, 173
                                                                                               method), 109
                      (AFL.automation.loading.Sensor.Sensor __init__() (AFL.automation.shared.MutableQueue.MutableQueue
__init__()
             method), 54, 173
                                                                                               method), 113, 114, 186
__init__() (AFL.automation.loading.SensorCallbackThreadinatelerConArilniaatimation.shared.PersistentConfig.MutableMapping
             method), 56, 70, 176
                                                                                               method), 115
__init__() (AFL.automation.loading.SensorCallbackThreadiSaintor(Q)l(Macdk.Tilutomation.shared.PersistentConfig.PersistentConfig
             method), 56, 57, 68, 174
                                                                                               method), 116-118, 187
__init__() (AFL.automation.loading.SensorCallbackThreadistinpleTDreAlfdldGBomation.shared.ServerDiscovery.AsyncServiceBrows
             method), 59, 60, 69, 176
                                                                                               method), 120, 121
__init__() (AFL.automation.loading.SensorCallbackThreadistate_LoadCBFL.automation.shared.ServerDiscovery.AsyncServiceInfo
             method), 62, 63, 68, 175
                                                                                               method), 122, 124
__init__() (AFL.automation.loading.SensorCallbackThreadistant_oddCBFL.automation.shared.ServerDiscovery.AsyncZeroconf
             method), 65, 66, 69, 175
                                                                                               method), 127
__init__() (AFL.automation.loading.SensorPollingThread_SiniorPollingFltzandomation.shared.ServerDiscovery.AsyncZeroconfServ
             method), 70, 71, 73, 177
                                                                                               method), 129
__init__() (AFL.automation.loading.SerialDevice.SerialDevinat__() (AFL.automation.shared.ServerDiscovery.IPVersion
             method), 74, 178
                                                                                               method), 130
__init__() (AFL.automation.loading.SyringePump.SyringePimipt_
                                                                                              () (AFL.automation.shared.ServerDiscovery.RunThread
             method), 75
                                                                                               method), 131, 149, 188
                      (AFL.automation.loading.Tubing __init__() (AFL.automation.shared.ServerDiscovery.ServerDiscovery
__init__()
                                                                                               method), 133, 134, 149, 189
             method), 76, 178
__init__() (AFL.automation.loading.UltimusVPressureContivolita;PtexsArleCantrollarion.shared.ServerDiscovery.ServiceBrowser
             method), 77
                                                                                               method), 135, 136
__init__() (AFL.automation.loading.UltimusVPressureContivition_UtinhAFVParusoumationtshadhed.ServerDiscovery.ServiceInfo
             method), 78, 79, 179
                                                                                               method), 139, 140
__init__() (AFL.automation.loading.ViciMultiposSelector.FimiSeleCip(AFL.automation.shared.ServerDiscovery.ServiceStateChange
             method), 79
                                                                                               method), 143
 __init__() (AFL.automation.loading.ViciMultiposSelecto<u>r.SariatDexGo</u>AFL.automation.shared.ServerDiscovery.Zeroconf
             method), 80
                                                                                               method), 144, 145
__init__() (AFL.automation.loading.ViciMultiposSelector_ViciMultif@\$Aflectortomation.shared.widgetui.Markdown
             method), 80, 81, 179
                                                                                               method), 155
__init__() (AFL.automation.shared.DataLabelerWidget.<u>DatirLebele</u>tMAIeL.automation.shared.ServerDiscovery.IPVersion
             method), 83, 84, 93, 182
                                                                                               class method), 130
 __init___() (AFL.automation.shared.DataLabelerWidget.<u>DatirLebele</u>CYteAFL.automation.shared.ServerDiscovery.ServiceStateChang
             method), 84, 93, 182
                                                                                               class method), 143
__init__() (AFL.automation.shared.DataLabelerWidget.<u>Dathenbel@) Willet</u>utomation.shared.ServerDiscovery.IPVersion
             method), 85, 86, 92, 181
                                                                                               class method), 130
__init__() (AFL.automation.shared.DataLabelerWidget.QrdimlEnQpdaFL.automation.shared.ServerDiscovery.ServiceStateChange
             method), 89, 90
                                                                                               class method), 143
__init__() (AFL.automation.shared.DatasetWidget.DatasetWidgetem__() (AFL.automation.shared.PersistentConfig.PersistentConfig
             method), 94, 95, 101, 182
                                                                                               method), 117, 119, 187
__init__() (AFL.automation.shared.DatasetWidget.DatasetWidget_Model
             method), 97, 102, 184
\_\_init\_\_() (AFL.automation.shared.DatasetWidget.DatasetWidget_aVi\notinV(AFL.automation.APIServer.QueueDaemon.DataTrashcan
             method), 98, 99, 102, 184
                                                                                               method), 11
\verb|\__init\__()| (AFL. automation. shared. Dataset Widget. default \verb|diet_listener()| (AFL. automation. shared. Server Discovery. Zero configuration of the property of the pro
             method), 99, 100
                                                                                               method), 148
__init__() (AFL.automation.shared.DiffractionLabeler.DiffractionLabeler(AFL.automation.shared.ServerDiscovery.AsyncZeroconf
             method), 103, 104, 112, 185
                                                                                               method), 129
__init__() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerMeden()
             method), 105, 112, 185
                                                                                               (AFL.automation.shared.ServerDiscovery.Zeroconf
__init__() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerMerMicpus
```

add_vertical_line() (AFL.automation.shared.DataLabelerWidget.DataL				
method), 84, 93, 182	AFL.automation.loading.FlowSelector			
add_vertical_line()	module, 34, 168			
	rakFionAurbehat/iem.loading.MultiChannelRelay			
method), 106, 113, 186	module, 35, 168			
Added (AFL.automation.shared.ServerDiscovery.ServiceSt attribute), 143	module, 36, 168			
${\tt addMode()}\ (AFL. automation. loading. Chemyx Syringe Pum_{plants})$	·			
method), 22, 26, 166	module, 40, 170			
	nd AFEIr view Logination.loading.PressureControllerAsPump			
attribute), 124	module, 41, 170			
addresses (AFL.automation.shared.ServerDiscovery.Serv				
attribute), 140	module, 44, 171			
addresses_by_version()	AFL.automation.loading.SampleCell			
(AFL.automation.shared.ServerDiscovery.Async				
method), 124	AFL.automation.loading.Sensor			
addresses_by_version()	module, 48, 172			
	e AFFb. automation.loading.SensorCallbackThread			
method), 140	module, 55, 174			
addX() (AFL.automation.loading.ChemyxSyringePump.Cl	•			
method), 22, 26, 166	module, 70, 176			
AFL.automation	AFL.automation.loading.SerialDevice			
module, 157, 193	module, 74, 177			
AFL.automation.APIServer	AFL.automation.loading.SyringePump			
module, 3, 158	module, 75, 178			
AFL.automation.APIServer.Client	AFL.automation.loading.Tubing			
module, 3, 158	module, 76, 178			
AFL.automation.APIServer.LoggerFilter	AFL.automation.loading.UltimusVPressureController			
module, 9, 159 AFL.automation.APIServer.QueueDaemon	module, 77, 178  AFL.automation.loading.ViciMultiposSelector			
module, 9, 160	module, 79, 179			
AFL.automation.EpicsADLiveProcess	AFL.automation.sample			
module, 160, 194	module, 82, 191			
AFL.automation.EpicsADLiveProcess.Client	AFL.automation.sample_env			
module, 161, 194	module, 82, 180			
AFL.automation.EpicsADLiveProcess.ReduceDaemo				
module, 161, 196	module, 83, 180			
AFL.automation.instrument	AFL.automation.shared.DataLabelerWidget			
module, 15, 162	module, 83, 180			
AFL.automation.instrument.FileCamera	AFL.automation.shared.DatasetWidget			
module, 15, 162	module, 94, 182			
AFL.automation.instrument.NetworkCamera	AFL.automation.shared.DiffractionLabeler			
module, 16, 162	module, 103, 185			
AFL.automation.loading	AFL.automation.shared.exceptions			
module, 17, 163	module, 150, 189			
AFL.automation.loading.CetoniMultiPosValve	AFL.automation.shared.MutableQueue			
module, 18, 163	module, 113, 186			
AFL.automation.loading.ChemyxSyringePump	AFL.automation.shared.PersistentConfig			
module, 20, 164	module, 114, 186			
AFL.automation.loading.DigitalOutPressureCont				
module, 26, 166	module, 151, 190			
AFL.automation.loading.DoubleViciMultiposSele	ecAT&Tr.automation.shared.ServerDiscovery			
module, 28, 167	module, 119, 188			

```
AFL.automation.shared.widgetui
                                                                                                                                                                                                      method), 147
             module, 152, 190
                                                                                                                                                                          async_remove_all_service_listeners()
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
aio_find_server_by_name()
                            (AFL.automation.APIServer.Client.ServerDiscovery
                                                                                                                                                                                                      method), 129
                           method), 7
                                                                                                                                                                          async_remove_listener()
aio_find_server_by_name()
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.Zeroconf
                            (AFL.automation.shared.ServerDiscovery.ServerDiscovery method), 148
                           method), 134, 149, 189
                                                                                                                                                                          async_remove_service_listener()
alive() (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.SensorPollingT
                            method), 71, 73, 177
                                                                                                                                                                                                      method), 129
All (AFL.automation.shared.ServerDiscovery.IPVersion async_request() (AFL.automation.shared.ServerDiscovery.AsyncService
                                                                                                                                                                                                      method), 124
                            attribute), 130
apply_isel() (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.DatasetW
                           method), 96, 102, 183
                                                                                                                                                                                                      method), 141
apply_isel() (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.DatasetW
                            method), 98, 102, 184
                                                                                                                                                                                                      method), 148
apply_sel() (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.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.D
                           method), 96, 102, 183
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
apply_sel() (AFL.automation.shared.DatasetWidget.DatasetWidget<u>m\text{Mthodell}</u>), 128
                           method), 98, 102, 184
                                                                                                                                                                          async_unregister_all_services()
async_add_listener()
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.Zeroconf
                           (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                      method), 147
                           method), 148
                                                                                                                                                                          async_unregister_service()
async_add_service_listener()
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
                           (AFL.automation.shared.ServerDiscovery.AsyncZeroconf method), 128
                           method), 128
                                                                                                                                                                          async_unregister_service()
async_cancel() (AFL.automation.shared.ServerDiscovery.AsyncSetAFdBautosantion.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                      method), 147
                           method), 122
                                                                                                                                                                          async_update_records()
async_check_service()
                           (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                      (AFL. automation. shared. Server Discovery. Async Service Browser
                           method), 148
                                                                                                                                                                                                      method), 122
async_clear_cache()
                                                                                                                                                                          async_update_records()
                            (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo(AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                                                                                                                                                                                                      method), 125
                           method), 124
async_clear_cache()
                                                                                                                                                                          async_update_records()
                           (AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                                                                                                                                      (AFL. automation. shared. Server Discovery. Service Browser\\
                                                                                                                                                                                                       method), 136
async_close() (AFL.automation.shared.ServerDiscovery.AsymcZarpdartfe_records()
                            method), 128
                                                                                                                                                                                                       (AFL.automation.shared.ServerDiscovery.ServiceInfo
async_find() (AFL.automation.shared.ServerDiscovery.AsyncZeroamefBord)iceTypes
                           class method), 129
                                                                                                                                                                          async_update_records_complete()
async_get_service_info()
                                                                                                                                                                                                       (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
                            (AFL.automation.shared.ServerDiscovery.AsyncZeroconf method), 122
                           method), 128
                                                                                                                                                                          async_update_records_complete()
async_get_service_info()
                                                                                                                                                                                                      (AFL. automation. shared. Server Discovery. Async Service Info
                            (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                      method), 125
                                                                                                                                                                          async_update_records_complete()
                           method), 147
async_notify_all() (AFL.automation.shared.ServerDiscovery.Zertalif.automation.shared.ServerDiscovery.ServiceBrowser
                            method), 146
                                                                                                                                                                                                      method), 137
                                                                                                                                                                          async_update_records_complete()
async_register_service()
                            (AFL.automation.shared.ServerDiscovery.AsyncZeroconf (AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                                                                                                                                      method), 141
                           method), 128
async_register_service()
                                                                                                                                                                          async_update_service()
                            (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                      (AFL.automation.shared.ServerDiscovery.AsyncZeroconf
```

method), 128	C
async_update_service()	calibrate() (AFL.automation.loading.Sensor.Sensor
(AFL.automation.shared.ServerDiscovery.Zeroco	onf method), 54, 173
method), 147	memoa), 54, 175
async_wait() (AFL.automation.shared.ServerDiscovery.A	camera_reset() (AFL.automation.instrument.NetworkCamera.NetworkCa AsyncServiceInfo_p_16_17_162
method), 125	
async_wait() (AFL.automation.shared.ServerDiscovery.S	cancel() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
method), 141	· · · · · · · · · · · · · · · · · · ·
	categories_(AFL.automation.shared.DataLabelerWidget.OrdinalEncode
async_wait() (AFL.automation.shared.ServerDiscovery.2	
method), 146	$\verb categories   (AFL. automation. shared. Diffraction Labeler. Ordinal Encoder and States and Stat$
async_wait_for_start()	attribute), 107
(AFL.automation.shared.ServerDiscovery.Zeroco	MetoniMultiPosValve (class in
method), 146	AFL. automation. loading. Cetoni Multi Pos Valve),
AsyncServiceBrowser (class in	18, 19, 164
AFL. automation. shared. Server Discovery),	<pre>change_model_callback()</pre>
120	(AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
AsyncServiceInfo (class in	method), 86, 93, 181
AFL.automation.shared.ServerDiscovery),	change_model_callback()
122	(AFL.automation.shared.DiffractionLabeler.DiffractionLabeler
AsyncZeroconf (class in	method), 104, 112, 185
AFL.automation.shared.ServerDiscovery),	change_norder_callback()
126	(AFL.automation.shared.DataLabelerWidget.DataLabelerWidget
AsyncZeroconfServiceTypes (class in	·
AFL.automation.shared.ServerDiscovery),	method), 86, 93, 181
129	change_norder_callback()
12)	(AFL.automation.shared.DiffractionLabeler.DiffractionLabeler
В	method), 104, 112, 185
	change_qstar_callback()
blockUntilStatusStopped()	(AFL. automation. shared. DataLabeler Widget. DataLabeler Widget)
(AFL. automation. loading. Chemyx Syringe Pump. Color of the property of the	ThemyxSyrupgaPunp86, 93, 181
method), 23, 25, 165	<pre>change_qstar_callback()</pre>
blockUntilStatusStopped()	(AFL. automation. shared. Diffraction Labeler. Diffraction Labeler
(AFL. automation. loading. Digital Out Pressure Continuous Conti	ntroller.DigitalQutP,ressureGontroller
method), 27	char_count() (AFL.automation.loading.UltimusVPressureController.Ulti
blockUntilStatusStopped()	method) 78 79 179
(AFL.automation.loading.DigitalOutPressureCon	venler_P145_spacesen(y \AFL.automation.APIServer.QueueDaemon.QueueD
method), 27	method), 13, 15, 160
blockUntilStatusStopped()	ChemyxConnection (class in
(AFL.automation.loading.DummyPump.DummyF	Pump AFL.automation.loading.ChemyxSyringePump),
method), 33, 34, 168	20, 25, 165
blockUntilStatusStopped()	ChemyxSyringePump (class in
	lkSyringePump),
method), 38, 40, 169	
blockUntilStatusStopped()	22, 24, 164
(AFL.automation.loading.PressureController.Pre	clear() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
method), 40, 41, 170	<i>''</i>
	clear() (AFL.automation.shared.DatasetWidget.defaultdict
blockUntilStatusStopped()	method), 100
(AFL.automation.ioaaing.PressureControllerAsP	une Bressur Contionation Smared Persistent Config. Mutable Mapping
method), 42, 44, 171	method), 115
blockUntilStatusStopped()	clear() (AFL.automation.shared.PersistentConfig.PersistentConfig
(AFL.automation.loading.UltimusVPressureCont	roller.Pressurg(Sartroller
method), 77	<pre>clear_history() (AFL.automation.APIServer.Client.Client</pre>
blockUntilStatusStopped()	method), 6, 8, 159
$(AFL. automation. loading. {\it Ultimus VP ressure Cont}$	relleg Letter Messure Controlletin module
method), 78	AFL.automation.shared.widgetui), 152

$\verb clear_queue()  (AFL. automation. APIS erver. Client. Clien$			d.ServerDiscovery.Se	rviceBrowser
method), 6, 8, 159		property), 137		
Client (class in AFL.automation.APIServer.Client), 3, 8,	DataLab	elerModel	(class	in
158		AFL.automation.share	ed.DataLabelerWidge	et),
${\tt Client} \ ({\it class in AFL. automation. Epics ADLive Process. Cliebully and the process. Cliebully and the process of the$	* *	83, 93, 181		
161, 194, 195	DataLab	elerView	(class	in
<pre>client_construct_ui()</pre>		AFL.automation.share	ed.DataLabelerWidge	et),
AFL.automation.shared.widgetui), 152, 156,		84, 93, 182		
191	DataLab	elerWidget	(class	in
${\tt close()}\ (AFL. automation. shared. Server Discovery. Zeroco. The state of the$	nf	AFL.automation.share	ed.DataLabelerWidge	et),
method), 148		85, 92, 181		
<pre>closeConnection() (AFL.automation.loading.ChemyxSy</pre>	rdnagteaPstenty	(AThleraythGroatioatishare	ed.DatasetWidget.Da	tasetWidget_Model
method), 21, 25, 165		property), 97, 102, 184	4	
collect() (AFL.automation.instrument.FileCamera.FileC	Dataset	Widget	(class	in
method), 16, 162		AFL.automation.share	ed.DatasetWidget),	
collect() (AFL.automation.instrument.NetworkCamera.N			0 /-	
method), 16, 17, 162		Widget_Model	(class	in
combine_vars() (AFL.automation.shared.DatasetWidget.	DatasetWi	ASEL.automation.share	ed.DatasetWidget),	
method), 97, 102, 183		97, 102, 184	0 //	
combine_vars() (AFL.automation.shared.DatasetWidget.			(class	in
method), 98, 102, 184		AFL.automation.share	*	
<pre>composition_click_callback()</pre>		98, 102, 184		
(AFL.automation.shared.DataLabelerWidget.Dat			(class	in
method), 86, 93, 181		AFL.automation.APIS	`	
composition_click_callback()	debug()		n.APIServer.Client.C	
(AFL.automation.shared.DatasetWidget.DatasetV		method), 6, 8, 159		
method), 96, 101, 183		_properties( <i>AFL.au</i>	tomation.shared.Ser	verDiscoverv.AsvncSei
compute_checksum() (AFL.automation.loading.UltimusV				
method), 78, 79, 179		_properties(AFL.au		verDiscovery.ServiceIi
copy() (AFL.automation.shared.DatasetWidget.defaultdict		attribute), 141		
method), 100		_factory( <i>AFL.autom</i>	ation shared Dataset	Widget defaultdict
		attribute), 100	on on one of the original of t	,, ragenaejammarer
D	default	* *	(class	in
daemon (AFL.automation.APIServer.QueueDaemon.Queue				
property), 13	Daemon	99	a.Baraser (rager);	
daemon (AFL.automation.EpicsADLiveProcess.ReduceDae	.denosia		on APIServer Client	Client
	intont.retiu	method), 6, 9, 159	omin i i ger ven emem.	Citciti
property), 197 daemon (AFL.automation.loading.Sensor.DummySensor1		* * * * * * * * * * * * * * * * * * * *	(in mo	dule
	ucscr1u.	AFL.automation.share	`	151,
property), 50		190	a.serianzanon),	131,
daemon (AFL.automation.loading.Sensor.DummySensor2	Diffrac	tionLabeler	(class	in
property), 53			\	
daemon (AFL.automation.loading.SensorCallbackThread.S	ensorCaiii	103, 112, 185	а.Біјјтаснопваосист	),
property), 58	·Di.ffrac		(class	in
daemon (AFL.automation.loading.SensorCallbackThread.S	i <del>mpte Frire</del>	AFL.automation.share	,	
property), 60			и.Біјјтиснопшиосист	),
${\tt daemon}(AFL. automation. loading. Sensor Callback Thread. Sensor Callbac$	Di ffrac	tionLabelerView	(class	in
property), 63				
${\tt daemon}(AFL. automation. loading. Sensor Callback Thread. Sensor Callbac$	topLoadC	<u>выг</u> л.ашотаноп.snare 105, 112, 185	и.Біјјиснопциоенег	),
property), 66	Di ai tal		ller ( <i>class</i>	in
daemon (AFL.automation.loading.SensorPollingThread.Sen	1 <del>501<b>4</b>-</del> 04 <del>1111</del> }	<b>ұққ<sub>ғайа</sub>ssurec</b> ontrol AFL.automation.loadi	na DiaitalOutDrassu	in reController)
property), 72		26, 28, 166	ng.DigitatOuti ressu	recommoner),
daemon (AFL.automation.shared.ServerDiscovery.RunThre	ad discove	zo,zo,100 r_server_by_name()	1	
property), 132	aracove.	(AFL.automation.APIS		``
		TARL dutomation API	Norvor   Hont Norvori	

```
method), 7
                                                                                 {\tt done}\,(AFL. automation. shared. Server Discovery. Async Service Browser
discover_server_by_name()
                                                                                               attribute), 122
             (AFL.automation.shared.ServerDiscovery.ServiceBrowser Lautomation.shared.ServerDiscovery.ServiceBrowser
             method), 134, 149, 189
                                                                                               attribute), 136
dispense() (AFL.automation.loading.ChemyxSyringePumpathhelpyitSyirInglePipugsSelector
                                                                                                                                          (class
             method), 23, 25, 165
                                                                                               AFL.automation.loading.DoubleViciMultiposSelector),
dispense() (AFL.automation.loading.ChemyxSyringePump.SyringePump1, 167
                                                                                 draw_peaks() (AFL.automation.shared.DataLabelerWidget.DataLabelerV
             method), 24
dispense() (AFL.automation.loading.DummyPump.DummyPump method), 86, 93, 181
                                                                                 draw_peaks() (AFL. automation. shared. Diffraction Labeler. Diffraction Lab
             method), 32, 34, 168
dispense() (AFL.automation.loading.DummyPump.SyringePump method), 104, 112, 185
                                                                                 driver_status() (AFL.automation.APIServer.Client.Client
             method), 34
dispense() (AFL.automation.loading.NE1kSyringePump.NE1kSyringedPhood), 5, 8, 158
             method), 37, 39, 169
                                                                                 driver_status() (AFL.automation.loading.Sensor.DummySensor2
dispense() (AFL.automation.loading.NE1kSyringePump.SyringePumpethod), 52, 55, 174
             method), 39
                                                                                 DummyPump (class in AFL.automation.loading.DummyPump),
dispense() (AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump
             method), 42, 44, 171
                                                                                 DummySensor1
                                                                                                                                                             in
dispense() (AFL:automation.loading.PressureControllerAsPump.SyAifikeAutomation.loading.Sensor),
                                                                                                                                                  49.
                                                                                                                                                           54.
dispense() (AFL.automation.loading.SyringePump.SyringDfmmySensor2
                                                                                                                              (class
                                                                                                                                                             in
             method), 75, 178
                                                                                               AFL.automation.loading.Sensor),
                                                                                                                                                           55,
dispenseRunning() (AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController
             method), 27
dispenseRunning() (AFL.automation.loading.DigitalOut FressureController.PressureController
             method), 28
                                                                                 Empty, 114
dispenseRunning() (AFL.automation.loading.PressureCompulger)PyreFluraGomerabarshared.MutableQueue.MutableQueue
             method), 40, 41, 170
                                                                                               method), 113, 114, 186
dispenseRunning() (AFL.automation.loading.UltimusVPEmptreMantpullonPteMuteController
             method), 77
                                                                                 emptySyringe() (AFL.automation.loading.ChemyxSyringePump.Chemyx
dispenseRunning() (AFL.automation.loading.UltimusVPressureContentlat)UttinusVPressureController
             method), 78
                                                                                 emptySyringe() (AFL.automation.loading.ChemyxSyringePump.SyringeF
display()
                                                                    module
                                                                                               method), 24
             AFL.automation.shared.widgetui), 152
                                                                                 emptySyringe() (AFL.automation.loading.DummyPump.DummyPump
dns_addresses() (AFL.automation.shared.ServerDiscovery.AsyncSenvihelinfo33, 34, 168
             method), 125
                                                                                 emptySyringe() (AFL.automation.loading.DummyPump.SyringePump
dns_addresses() (AFL.automation.shared.ServerDiscovery.ServiceInfehod), 34
             method), 141
                                                                                 emptySyringe() (AFL.automation.loading.NE1kSyringePump.NE1kSyrin
dns_nsec() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfethod), 38, 40, 169
             method), 125
                                                                                 emptySyringe() (AFL.automation.loading.NE1kSyringePump.SyringePun
dns_nsec() (AFL.automation.shared.ServerDiscovery.ServiceInfo method), 39
             method), 141
                                                                                 \verb"emptySyringe" () \ (AFL. automation. loading. Pressure Controller As Pump. Pressure Controller As P
dns_pointer() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfd), 42, 44, 171
             method), 125
                                                                                 emptySyringe() (AFL.automation.loading.PressureControllerAsPump.Sy
dns_pointer() (AFL.automation.shared.ServerDiscovery.ServiceInfmethod), 43
             method), 141
                                                                                  emptySyringe() (AFL.automation.loading.SyringePump.SyringePump
dns_service() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfd), 75, 178
             method), 125
                                                                                 enqueue()
                                                                                                       (AFL.automation.APIServer.Client.Client
dns_service() (AFL.automation.shared.ServerDiscovery.ServiceInfmethod), 5, 8, 159
             method), 141
                                                                                 enqueued_base() (AFL.automation.APIServer.Client.Client
dns_text() (AFL.automation.shared.ServerDiscovery.AsyncServiceInfethod), 5, 8, 158
             method), 125
                                                                                 extract_var() (AFL.automation.shared.DatasetWidget.DatasetWidget
dns_text() (AFL.automation.shared.ServerDiscovery.ServiceInfo method), 97, 102, 183
             method), 141
```

```
extract_var() (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.Dataset
                              method), 98, 102, 184
                                                                                                                                                                                                                 class method), 100
                                                                                                                                                                                  Full, 114
F
feature_names_in_(AFL.automation.shared.DataLabelerwidget.OrdinalEncoder
                                                                                                                                                                                  generate_service_broadcast()
                              attribute), 88
 feature_names_in_(AFL.automation.shared.DiffractionLabeler.OrtMAEdlEnutothation.shared.ServerDiscovery.Zeroconf
                              attribute), 107
                                                                                                                                                                                                                method), 147
FileCamera (class in AFL.automation.instrument.FileCameganerate_service_query()
                                                                                                                                                                                                                (AFL.automation.shared.ServerDiscovery.Zeroconf
finalize() (AFL.automation.APIServer.QueueDaemon.DataTrashcomethod), 147
                                                                                                                                                                                  generate_unregister_all_services()
                              method), 11
find() (AFL:automation.shared.ServerDiscovery.AsyncZeroconfServiAfIlypestomation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                                 method), 147
                              class method), 129
find_server_by_name()
                                                                                                                                                                                  get() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
                             (AFL.automation.APIServer.Client.ServerDiscovery
                                                                                                                                                                                                                method), 11
                                                                                                                                                                                  get() (AFL.automation.shared.DatasetWidget.defaultdict
                             method), 7
find_server_by_name()
                                                                                                                                                                                                                method), 100
                              (AFL.automation.shared.ServerDiscovery.ServerDiscovery.Fl.automation.shared.MutableQueue.MutableQueue
                             method), 134, 149, 189
                                                                                                                                                                                                                method), 113, 114, 186
                                                                                                                                                                                  get() (AFL.automation.shared.PersistentConfig.MutableMapping
find_server_by_partial_name()
                              (AFL.automation.APIServer.Client.ServerDiscovery
                                                                                                                                                                                                                method), 115
                             method), 7
                                                                                                                                                                                   get() (AFL.automation.shared.PersistentConfig.PersistentConfig
find_server_by_partial_name()
                                                                                                                                                                                                                method), 118
                              (AFL.automation.shared.ServerDiscovery.ServerDiscoveryaddress_and_nsec_records()
                                                                                                                                                                                                                 (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                             method), 134, 150, 189
find_server_by_property_match()
                                                                                                                                                                                                                method), 125
                             (AFL.automation.APIServer.Client.ServerDiscoveget_address_and_nsec_records()
                                                                                                                                                                                                                (AFL.automation.shared.ServerDiscovery.ServiceInfo
                             method), 7
find_server_by_property_match()
                                                                                                                                                                                                                method), 142
                             (AFL.automation.shared.ServerDiscovery.ServerDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDiscoveryDis
                                                                                                                                                                                                                method), 98, 102, 184
                             method), 134, 150, 189
\verb|fit()| (AFL. automation. shared. DataLabeler Widget. Ordinal \textit{General Comps}())| (AFL. automation. shared. Dataset Widget. Dataset. 
                             method), 90
                                                                                                                                                                                                                method), 96, 102, 183
\verb|fit()| (AFL. automation. shared. Diffraction Labeler. Ordinal \textit{Egetodeon} \verb|fig()| (AFL. automation. APIS erver. Client. 
                              method), 109
                                                                                                                                                                                                                method), 5, 8, 159
fit_transform() (AFL.automation.shared.DataLabelerWightetDrivealEnhipelet()
                                                                                                                                                                                                                 (AFL.automation.APIServer.Client.Client
                              method), 91
fit_transform() (AFL.automation.shared.DiffractionLabeler.Ordinal Monaduler, 9, 159
                             method), 110
                                                                                                                                                                                  get_feature_names_out()
FlowSelector
                                                                                                 (class
                                                                                                                                                                                                                 (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
                                                                                                                                                                     in
                             AFL.automation.loading.CetoniMultiPosValve),
                                                                                                                                                                                                                method), 91
                                                                                                                                                                                  get_feature_names_out()
                                                                                                                                                                                                                (AFL. automation. shared. Diffraction Labeler. Ordinal Encoder\\
FlowSelector
                                                                                                 (class
                                                                                                                                                                      in
                             AFL.automation.loading.DoubleViciMultiposSelector),
                                                                                                                                                                                                                method), 110
                                                                                                                                                                                  get_historical_values()
FlowSelector
                                                                                                 (class
                                                                                                                                                                     in
                                                                                                                                                                                                                (AFL.automation.shared.PersistentConfig.PersistentConfig
                             AFL.automation.loading.FlowSelector),
                                                                                                                                                                                                                method), 118, 119, 188
                              34, 35, 168
                                                                                                                                                                                  get_metadata_routing()
FlowSelector
                                                                                                                                                                                                                (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
                                                                                                 (class
                                                                                                                                                                     in
                                                                                                                                                                                                                method), 91
                             AFL.automation.loading.ViciMultiposSelector),
                                                                                                                                                                                   get_metadata_routing()
from_server_name() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
                             class method), 5, 8, 158
                                                                                                                                                                                                                method), 111
```

```
get_name() (AFL.automation.shared.ServerDiscovery.AsyngeSeName())f(AFL.automation.loading.Sensor.DummySensor1
                                   method), 125
                                                                                                                                                                                                                                                                method), 50
get_name() (AFL.automation.shared.ServerDiscovery.ServgetMame() (AFL.automation.loading.Sensor.DummySensor2
                                   method), 142
                                                                                                                                                                                                                                                                method), 53
get_non_sample_dims()
                                                                                                                                                                                                                           getName() (AFL. automation. loading. Sensor Callback Thread. Sensor Callbac
                                   (AFL.automation.shared.DatasetWidget.DatasetWidget_Modeethod), 58
                                   method), 98, 102, 184
                                                                                                                                                                                                                           getName() (AFL.automation.loading.SensorCallbackThread.SimpleThresh
get_object() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                                method), 60
                                   method), 6, 9, 159
                                                                                                                                                                                                                           getName() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv
get_params()(AFL.automation.shared.DataLabelerWidget.OrdinalEnathdeln), 63
                                   method), 91
                                                                                                                                                                                                                           \verb"getName" () (AFL. automation. loading. Sensor Callback Thread. Stop Load CB with the control of the control
get_params() (AFL.automation.shared.DiffractionLabeler.OrdinalEmathod), 66
                                                                                                                                                                                                                           \verb"getName" () (AFL. automation. loading. Sensor Polling Thread. Se
                                   method), 111
get_peaks() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWi
                                   method), 84, 93, 182
                                                                                                                                                                                                                           {\tt getName()} \ (AFL. automation. shared. Server Discovery. Run Thread
\verb|get_peaks()| (AFL. automation. shared. Diffraction Labeler. Diffraction \textit{hathada}) | \verb|Model|| | AFL. automation. shared. Diffraction \textit{Labeler. Diffraction}| | AFL. automation. shared. Diffraction. Shared. Diffraction. Shared. Diffraction. Shared. 
                                                                                                                                                                                                                           \verb"getName" () (AFL. automation. shared. Server Discovery. Service Browser"
                                   method), 105, 112, 185
get_queue() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                               method), 137
                                   method), 5, 8, 158
                                                                                                                                                                                                                           getOpenPorts()
                                                                                                                                                                                                                                                                                                                                                                                                                  module
get_queued_commands()
                                                                                                                                                                                                                                                               AFL.automation.loading.ChemyxSyringePump),
                                   (AFL. automation. API Server. Client. Client
                                                                                                                                                                                                                                                                20, 25, 165
                                   method), 5, 8, 159
                                                                                                                                                                                                                           getParameterLimits()
get_quickbar() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                                (AFL.automation.loading.ChemyxSyringePump.ChemyxConnecti
                                                                                                                                                                                                                                                                method), 22, 25, 166
                                   method), 5, 8, 158
get_scattering() (AFL.automation.shared.DatasetWidgegetParasetWidges (W & FL.automation.loading.ChemyxSyringePump.Chemy.
                                   method), 98, 102, 184
                                                                                                                                                                                                                                                               method), 22, 26, 166
get_server_time() (AFL.automation.APIServer.Client.CljertPort() (AFL.automation.loading.CetoniMultiPosValve.CetoniMultiPos
                                                                                                                                                                                                                                                                method), 19, 164
                                   method), 5, 8, 159
get_service_info() (AFL.automation.APIServer.Client.SpetPcDtsC) (&FL.automation.loading.CetoniMultiPosValve.FlowSelector
                                   method), 7
                                                                                                                                                                                                                                                                method), 19
get_service_info() (AFL.automation.shared.ServerDisagerPoSta()e(AFSLautomation.loading.DoubleViciMultiposSelector.DoubleV
                                   method), 134, 149, 189
                                                                                                                                                                                                                                                                method), 29, 31, 167
get_service_info() (AFL.automation.shared.ServerDisagetPoZtrOc(AFL.automation.loading.DoubleViciMultiposSelector.FlowSel
                                   method), 146
                                                                                                                                                                                                                                                                method), 29
get_unqueued_commmands()
                                                                                                                                                                                                                           getPort() (AFL.automation.loading.DoubleViciMultiposSelector.ViciMult
                                    (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                               method), 31
                                   method), 5, 8, 158
                                                                                                                                                                                                                           getPort() (AFL.automation.loading.FlowSelector.FlowSelector
getChannels() (AFL.automation.loading.MultiChannelRelay.MultiGhethoellRelay168
                                    method), 35, 36, 168
                                                                                                                                                                                                                           {\tt getPort()} \ (AFL. automation. loading. ViciMultipos Selector. Flow Selector
getChannels() (AFL.automation.loading.SainSmartRelay.MultiChamethRelay,80
                                   method), 45
                                                                                                                                                                                                                           getPort() (AFL.automation.loading.ViciMultiposSelector.ViciMultiposSel
getChannels() (AFL.automation.loading.SainSmartRelay.SainSmartRelayd), 81, 82, 180
                                                                                                                                                                                                                           getPumpStatus() (AFL.automation.loading.ChemyxSyringePump.Chemy.
                                   method), 46, 47, 171
getDisplacedVolume()
                                                                                                                                                                                                                                                               method), 22, 26, 166
                                   (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.ChemyxSyringePump.Chemyx
                                   method), 22, 26, 166
                                                                                                                                                                                                                                                                method), 23, 25, 165
getElapsedTime() (AFL.automation.loading.ChemyxSyringertRanp.QhenFlxGutmactiiom.loading.ChemyxSyringePump.SyringePump
                                   method), 22, 26, 166
                                                                                                                                                                                                                                                                method), 24
getLevel() (AFL.automation.loading.ChemyxSyringePumgatRangeSyringePumpmation.loading.DummyPump.DummyPump
                                    method), 23, 25, 165
                                                                                                                                                                                                                                                                method), 33, 34, 168
getName() (AFL.automation.APIServer.QueueDaemon.QuegetRactneth) (AFL.automation.loading.DummyPump.SyringePump
                                                                                                                                                                                                                                                               method), 34
getName() (AFL.automation.EpicsADLiveProcess.ReduceDgetRart ReduAHDacentomation.loading.NE1kSyringePump.NE1kSyringePump
```

*method*), 197

method), 38, 39, 169

```
getRate() (AFL.automation.loading.NE1kSyringePump.Syringer(Mrf)L.automation.loading.SensorCallbackThread.SimpleThresholdC.
                                method), 39
                                                                                                                                                                                                                                 property), 60
getRate() (AFL.automation.loading.PressureControllerAsPdent PAdSkuru@omatidheldadfungSensorCallbackThread.StopLoadCBv1
                                method), 42, 44, 171
                                                                                                                                                                                                                                 property), 63
getRate() (AFL.automation.loading.PressureControllerAsPdverpt.SAHillgaRumpution.loading.SensorCallbackThread.StopLoadCBv2
                                method), 43
                                                                                                                                                                                                                                 property), 66
getRate() (AFL.automation.loading.SyringePump.SyringePdent (AFL.automation.loading.SensorPollingThread.SensorPollingThread
                                method), 75, 178
                                                                                                                                                                                                                                 property), 72
getResponse() (AFL.automation.loading.ChemyxSyringeHudeptClAeFilyxQttomatitionshared.ServerDiscovery.RunThread
                                method), 21, 25, 165
                                                                                                                                                                                                                                 property), 132
getServerState() (AFL.automation.loading.SensorCallbixdeFtreAdF.LoandonCoinmsdmiaratisServerDiscovery.ServiceBrowser
                                method), 56, 70, 176
                                                                                                                                                                                                                                  property), 137
getStatus() (AFL.automation.loading.ChemyxSyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyringePunip_EdeapurerStyr
                                                                                                                                                                                                                                  (AFL.automation.shared.DataLabelerWidget.OrdinalEncoder
                                method), 23, 25, 165
getStatus() (AFL.automation.loading.DummyPump.DummyPump attribute), 88
                                method), 33, 34, 168
                                                                                                                                                                                                  infrequent_categories_
getStatus() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePumpomation.shared.DataLabelerWidget.OrdinalEncoder
                                method), 38, 40, 169
                                                                                                                                                                                                                                 property), 92
getStatus() (AFL.automation.loading.PressureControllerArtProopsProtestatus()
                                method), 42, 44, 171
                                                                                                                                                                                                                                  (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
getValueFromParams()
                                                                                                                                                                                                                                  attribute), 107
                                (AFL.automation.loading.ChemyxSyringePump.ChevfiveSyvengePartpgories_
                                method), 23, 25, 165
                                                                                                                                                                                                                                  (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
goto_callback() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabelerWidget.DataLabel
                                                                                                                                                                                                  init_buttons() (AFL.automation.shared.DatasetWidget_DatasetWidget_
                                method), 86, 93, 181
goto_callback() (AFL.automation.shared.DatasetWidget.DatasetWindghtod), 99, 103, 184
                                method), 96, 101, 183
                                                                                                                                                                                                  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.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Dataset.Da
goto_callback() (AFL.automation.shared.DiffractionLabeler.Diffrantithold)b0ler103, 184
                                method), 104, 112, 185
                                                                                                                                                                                                  init_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget
                                                                                                                                                                                                                                  method), 99, 103, 184
Η
                                                                                                                                                                                                  init_inputs() (AFL.automation.shared.DatasetWidget.DatasetWidget_V
                                                                                                                                                                                                                                  method), 99, 103, 184
                                                   (AFL.automation.APIServer.Client.Client
halt()
                                                                                                                                                                                                  init_models() (AFL.automation.shared.DataLabelerWidget.DataLabeler
                                method), 6, 8, 159
                                                                                                                                                                                                                                  method), 84, 93, 182
halt() (AFL.automation.EpicsADLiveProcess.Client.Client
                                                                                                                                                                                                  init_models() (AFL.automation.shared.DiffractionLabeler.DiffractionLa
                                method), 161, 195, 196
                                                                                                                                                                                                                                  method), 105, 112, 185
home() (AFL.automation.EpicsADLiveProcess.Client.Client
                                                                                                                                                                                                  init_plots() (AFL.automation.shared.DatasetWidget.DatasetWidget Vie
                                 method), 161, 195, 196
host_ttl (AFL.automation.shared.ServerDiscovery.AsyncServiceInfonethod), 99, 103, 184
                                                                                                                                                                                                  initialize_plots() (AFL.automation.shared.DatasetWidget.DatasetWid
                                attribute), 124
                                                                                                                                                                                                                                  method), 96, 102, 183
\verb|host_ttl| (AFL. automation. shared. Server Discovery. Service Info
                                                                                                                                                                                                  interface_index(AFL.automation.shared.ServerDiscovery.AsyncService
                                attribute), 140
                                                                                                                                                                                                                                  attribute), 124
                                                                                                                                                                                                 interface_index(AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                                                                                                                                                                  attribute), 140
ident (AFL.automation.APIServer.QueueDaemon.QueueDaemon
                                                                                                                                                                                                  inverse_transform()
                                property), 13
\textbf{ident} (AFL. automation. Epics ADLive Process. Reduce Daemon. 
                                                                                                                                                                                                                                  method), 90
                                property), 197
                                                                                                                                                                                                  inverse_transform()
ident (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                                  (AFL. automation. shared. Diffraction Labeler. Ordinal Encoder\\
                                property), 50
                                                                                                                                                                                                                                  method), 110
ident (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                  ip_addresses_by_version()
                                property), 53
\textbf{ident} \ (AFL. automation. loading. Sensor Callback Thread. Sensor Callback Thread} \ (AFL. automation. shared. Server Discovery. As ync Service Information and the sensor Callback Thread Sensor Callbac
                                                                                                                                                                                                                                  method), 125
                                property), 58
                                                                                                                                                                                                  ip_addresses_by_version()
```

```
(AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                                                                                                                                                                                                                                                       method), 11
                                            method), 142
                                                                                                                                                                                                                                                                           items() (AFL.automation.shared.DatasetWidget.defaultdict
IPVersion (class in AFL.automation.shared.ServerDiscovery),
                                                                                                                                                                                                                                                                                                                        method), 100
                                                                                                                                                                                                                                                                          items() (AFL.automation.shared.PersistentConfig.MutableMapping
is_alive() (AFL.automation.APIServer.QueueDaemon.QueueDaemonthod), 115
                                                                                                                                                                                                                                                                          items() (AFL.automation.shared.PersistentConfig.PersistentConfig
                                            method), 13
is_alive() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.RectionsDaethon
                                             method), 198
                                                                                                                                                                                                                                                                           iterationid() (AFL.automation.shared.MutableQueue.MutableQueue
is_alive() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                                                                                                                        method), 113, 114, 186
                                              method), 50
is_alive() (AFL.automation.loading.Sensor.DummySensor)2
                                              method), 53
                                                                                                                                                                                                                                                                            join() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
is_alive() (AFL.automation.loading.SensorCallbackThread.SensorGallbackThread
                                             method), 58
                                                                                                                                                                                                                                                                            join() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDae
is_alive()(AFL.automation.loading.SensorCallbackThread.SimpleThreshpldGB
                                              method), 60
                                                                                                                                                                                                                                                                            join() (AFL.automation.loading.Sensor.DummySensor1
is_alive() (AFL.automation.loading.SensorCallbackThread.StopLoadGByl, 50
                                             method), 63
                                                                                                                                                                                                                                                                           join() (AFL.automation.loading.Sensor.DummySensor2
is_alive() (AFL.automation.loading.SensorCallbackThread.StopLoad GBy 3, 53
                                             method), 66
                                                                                                                                                                                                                                                                            join() (AFL.automation.loading.SensorCallbackThread.SensorCallbackT
is_alive() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
                                            method), 72
                                                                                                                                                                                                                                                                            join() (AFL.automation.loading.SensorCallbackThread.SimpleThreshold
is_alive() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                                                                                                                      method), 61
                                              method), 132
                                                                                                                                                                                                                                                                           join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
is_alive() (AFL.automation.shared.ServerDiscovery.ServiceBrowsquethod), 64
                                            method), 137
                                                                                                                                                                                                                                                                           join() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
is_serialized()
                                                                                                                                                                                                                                module
                                                                                                                                                     (in
                                                                                                                                                                                                                                                                                                                        method), 67
                                            AFL.automation.APIServer.QueueDaemon), 10
                                                                                                                                                                                                                                                                           join() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
is_serialized()
                                                                                                                                                                                                                                module
                                                                                                                                                     (in
                                                                                                                                                                                                                                                                                                                        method), 72
                                                                                                                                                                                                                                             151,
                                            AFL.automation.shared.serialization),
                                                                                                                                                                                                                                                                           join() (AFL.automation.shared.ServerDiscovery.RunThread
                                              190
                                                                                                                                                                                                                                                                                                                        method), 132
is {\tt Daemon()}\ (AFL. automation. APIS erver. Queue Daemon. Queue {\tt Daemon.}\ Queue {
                                            method), 13
                                                                                                                                                                                                                                                                                                                        method), 137
isDaemon() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon
                                             method), 198
\verb|isDaemon()| (AFL. automation. loading. Sensor. Dummy Sensor!_{\texttt{Key}} (AFL. automation. shared. Server Discovery. A sync Service Information. Sensor. Dummy Sensor!_{\texttt{New}} (AFL. automation. shared. Server Discovery. A sync Service Information. Sensor. Dummy Senso
                                            method), 50
                                                                                                                                                                                                                                                                                                                        attribute), 124
\verb|isDaemon()| (AFL. automation. loading. Sensor. Dummy Sensor_2 \\ \verb|key(AFL. automation. shared. Server Discovery. Service Infonctional Content of the Con
                                             method), 53
                                                                                                                                                                                                                                                                                                                        attribute), 140
{\tt isDaemon()} \ (AFL. automation. loading. Sensor Callback Thread Sensor Callback Thread Revs() (AFL. automation. API Server. Queue Daemon. Data Trash can be a support of the contraction of the contra
                                            method), 58
                                                                                                                                                                                                                                                                                                                        method), 1
isDaemon() (AFL.automation.loading.SensorCallbackThread,SimpleThrealfoldCB, (AFL.automation.shared.DatasetWidget.defaultdict
                                            method), 60
                                                                                                                                                                                                                                                                                                                        method), 100
is {\tt Daemon()} \ (AFL. automation. loading. Sensor Callback Thread, Stop Load CBv1, automation. shared. Persistent Config. Mutable Mapping (AFL. automation. shared. Persistent Config. Mutable Mapping Config. Mutable Mapping (AFL. automation. shared. Persistent Config. Mutable Mapping (AFL. automation. shared. Sha
                                            method), 63
                                                                                                                                                                                                                                                                                                                        method), 115
\verb|isDaemon()| (AFL. automation. loading. Sensor Callback Thread Stop Load CBy 2 (AFL. automation. shared. Persistent Config. 
                                             method), 66
isDaemon() (AFL.automation.loading.SensorPollingThread.SensorPollingThread
                                             method), 72
isDaemon() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                                                                                                                     (AFL. automation. shared. Data Labeler Widget. Data Labeler Widget (AFL. automation. shared. Data Labeler Widget) and the property of the pr
                                            method), 132
isDaemon() (AFL.automation.shared.ServerDiscovery.ServiceBrowser method), 86, 93, 181 method), 137
                                                                                                                                                                                                                                                                                                                     (AFL. automation. shared. Diffraction Labeler. Diffraction Labeler)
```

items() (AFL.automation.APIServer.QueueDaemon.DataTrashcan

method), 104, 112, 185

listeners (AFL.automation.shared.ServerDiscovery.Zeroconf	(AFL.automation.shared.ServerDiscovery.ServerDiscovery		
property), 146	method), 134, 149, 189		
load_from_cache() (AFL.automation.shared.ServerDiscaments	•		
method), 125	attribute), 155		
load_from_cache() (AFL.automation.shared.ServerDiscolling method), 142 mod	ule		
load_instrument() (AFL.automation.EpicsADLiveProcess.C	AFL.automation.APIServer, 3, 158		
method), 161, 195, 196 load_labware() (AFL.automation.EpicsADLiveProcess.Clien			
method), 161, 195, 196	AFL.automation.APIServer.LoggerFilter, 9,		
LoaderCommunication (class in	159		
AFL.automation.loading.SensorCallbackThread),	AFL.automation.APIServer.QueueDaemon, 9,		
56, 70, 176	160		
${\tt loadSample()} \ (AFL. automation. loading. Sample Cell. Sample Ce$			
method), 48, 172	194		
<pre>log_exception_debug()</pre>	AFL.automation.EpicsADLiveProcess.Client,		
(AFL. automation. shared. Server Discovery. Zero configure 100% and 100% are also below the property of the	161, 194		
class method), 148	AFL.automation.EpicsADLiveProcess.ReduceDaemon,		
log_exception_once()	161, 196		
(AFL.automation.shared.ServerDiscovery.Zeroconf	AFL.automation.instrument, 15, 162		
class method), 148	AFL.automation.instrument.FileCamera, 15,		
<pre>log_exception_warning()</pre>	162		
(AFL.automation.shared.ServerDiscovery.Zeroconf	AFL.automation.instrument.NetworkCamera,		
class method), 148	16, 162		
log_warning_once() (AFL.automation.shared.ServerDiscove			
class method), 148	AFL.automation.loading.CetoniMultiPosValve,		
logged_in() (AFL.automation.APIServer.Client.Client	18, 163		
method), 5, 8, 158	AFL.automation.loading.ChemyxSyringePump,		
logged_in() (AFL.automation.EpicsADLiveProcess.Client.Cli			
method), 161, 195	AFL.automation.loading.DigitalOutPressureController,		
LoggerFilter (class in	26, 166		
AFL.automation.APIServer.LoggerFilter), 9, 159	AFL.automation.loading.DoubleViciMultiposSelector, 28,167		
login() (AFL.automation.APIServer.Client.Client	AFL.automation.loading.DummyPump, 32, 167		
method), 5, 8, 158	AFL.automation.loading.FlowSelector, 34,		
login() (AFL.automation.EpicsADLiveProcess.Client.Client	168		
method), 161, 195	AFL.automation.loading.MultiChannelRelay,		
	35, 168		
M	AFL.automation.loading.NE1kSyringePump,		
<pre>manage_service_info_to_list()</pre>	36, 168		
(AFL.automation.APIServer.Client.ServerDiscovery	AFL.automation.loading.PressureController,		
method), 7	40, 170		
<pre>manage_service_info_to_list()</pre>	AFL.automation.loading.PressureControllerAsPump,		
(AFL. automation. shared. Server Discovery. Server Discovery and the property of the propert	overy 41, 170		
method), 134, 149, 189	AFL.automation.loading.SainSmartRelay,		
Markdown (class in AFL.automation.shared.widgetui),	44, 171		
155	AFL.automation.loading.SampleCell, 48, 172		
mask_serialized_objs()	AFL automation loading Sensor, 48, 172		
	neAFIn.automation.loading.SensorCallbackThread,		
method), 14, 15, 160	55, 174		
match_server_by_name()	AFL.automation.loading.SensorPollingThread, 70,176		
(AFL.automation.APIServer.Client.ServerDiscovery	AFL.automation.loading.SerialDevice, 74,		
<pre>method), 7 match server by name()</pre>	177		
march server by namell			

AFL.automation.loading.SyringePump, 75, 178	name (AFL.automation.loading.Sensor.DummySensor2 property), 53
AFL.automation.loading.Tubing, 76, 178	$\verb name  (AFL. automation. loading. Sensor Callback Thread. Sensor Callback T$
AFL.automation.loading.UltimusVPressureCon 77,178	
AFL.automation.loading.ViciMultiposSelector	· · · · · · · · · · · · · · · · · · ·
79, 179	name (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
AFL.automation.sample, 82, 191	property), 64
AFL.automation.sample_env, 82, 180	name (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
AFL.automation.shared, 83, 180	property), 67
AFL.automation.shared.DataLabelerWidget,	name (AFL.automation.loading.SensorPollingThread.SensorPollingThread
83, 180	property), 72
AFL.automation.shared.DatasetWidget, 94,	name (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
182	attribute), 125
	name (AFL.automation.shared.ServerDiscovery.RunThread
103, 185	property), 133
AFL automation.shared.exceptions, 150, 189	name (AFL.automation.shared.ServerDiscovery.ServiceBrowser
AFL.automation.shared.MutableQueue, 113,	property), 137
186	name (AFL.automation.shared.ServerDiscovery.ServiceInfo
AFL.automation.shared.PersistentConfig,	attribute), 142
114, 186	native_id(AFL.automation.APIServer.QueueDaemon.QueueDaemon
AFL.automation.shared.serialization, 151,	property), 14
190	$\verb"native_id" (AFL. automation. Epics ADLive Process. Reduce Daemon. Reduce Daemon. The approximation of the process of the p$
AFL.automation.shared.ServerDiscovery,	property), 198
119, 188	<pre>native_id(AFL.automation.loading.Sensor.DummySensor1</pre>
AFL.automation.shared.widgetui, 152, 190	property), 51
	umative_id(AFL.automation.loading.Sensor.DummySensor2
method), 114, 186	property), 53
	$\verb"native_id" (AFL. automation. loading. Sensor Callback Thread. Sensor Callb$
method), 6, 9, 159	property), 58
MultiChannelRelay (class in	$\verb"native_id" (AFL. automation. loading. Sensor Callback Thread. Simple Threst Callback Thread. Simple Threst Callback Thread. Simple Thread$
AFL. automation. loading. Multi Channel Relay),	property), 61
35, 168	$\verb"native_id" (AFL. automation. loading. Sensor Callback Thread. Stop Load Change Control of the Control of th$
MultiChannelRelay (class in	property), 64
AFL.automation.loading.SainSmartRelay), 45	<pre>native_id(AFL.automation.loading.SensorCallbackThread.StopLoadCl property), 67</pre>
MutableMapping (class in	$\verb"native_id" (AFL. automation. loading. Sensor Polling Thread. Sen$
AFL.automation.shared.PersistentConfig),	property), 72
115	native_id(AFL.automation.shared.ServerDiscovery.RunThread
MutableQueue (class in	property), 133
AFL.automation.shared.MutableQueue),	native_id(AFL.automation.shared.ServerDiscovery.ServiceBrowser
113, 114, 186	property), 138
110, 11 ., 100	NE1kSyringePump (class in
N	AFL.automation.loading.NE1kSyringePump),
${\tt n\_features\_in\_(AFL. automation. shared. DataLabeler Weattribute), 88}$	NetworkCamera (class in
n_features_in_(AFL.automation.shared.DiffractionLaborattribute), 107	eler.OrdinaÆheoutomation.instrument.NetworkCamera), 16, 162
name (AFL.automation.APIServer.QueueDaemon.QueueDa	
property), 14	(AFL.automation.shared.DataLabelerWidget.DataLabelerWidge
name (AFL.automation.EpicsADLiveProcess.ReduceDaemo	
property), 198	next_button_callback()
name (AFL.automation.loading.Sensor.DummySensor1	(AFL.automation.shared.DatasetWidget.DatasetWidget
property), 51	method), 96, 101, 183

property), 51

$\label{lem:callback} \mbox{next\_button\_callback()} \\ (AFL. automation. shared. Diffraction Labeler. Diffraction $			(class PersistentConfig),	in
method), 104, 112, 185		116, 118, 187		
NoDeviceFoundException, 150, 151, 190 NotFoundError, 150, 151, 190		mp() (AFL.automation.s. method), 99, 103, 184	hared.DatasetWidget	.DatasetWidget_Viev
notify_all() (AFL.automation.shared.ServerDiscovery.2 method), 146		s() (AFL.automation.sha method), 99, 103, 184	ared.DatasetWidget.l	DatasetWidget_View
0		FL.automation.APIServe method), 11	r.QueueDaemon.Da	taTrashcan
on_service_state_change()		FL.automation.shared.D	atasetWidget.default	dict
(AFL.automation.APIServer.Client.ServerDiscove		method), 100	0 ,	
method), 7		FL.automation.shared.Pe	ersistentConfig.Muta	bleMapping
on service state change()		method), 115		
(AFL.automation.shared.ServerDiscovery.Server	DASPONE A	FL.automation.shared.Pe	ersistentConfig.Persis	stentConfig
method), 134, 149, 189		method), 118		
openConnection() (AFL.automation.loading.ChemyxSyrumethod), 21, 25, 165	ingeritan.	OHAFJx <b>a</b> stanvation.API method), 11	Server.QueueDaemo	n.DataTrashcan
ordinal_phase_labels()		() (AFL.automation.sha	red.DatasetWidget.de	efaultdict
(AFL. automation. shared. Data Labeler Widget. Data Control of the control of t	aLabelerN	prathod), 100		
method), 84, 93, 182	popitem	() (AFL.automation.sha	red.PersistentConfig	MutableMapping
ordinal_phase_labels()		method), 115		
(AFL.automation.shared.DiffractionLabeler.Diffraction), 105, 112, 185		method), 118		
(		L.automation.shared.Ser	verDiscovery.AsyncS	ServiceInfo
AFL. automation. shared. Data Labeler Widget),		attribute), 124	D:	T C
86		L.automation.shared.Ser	verDiscovery.Servic	eInfo
OrdinalEncoder (class in		attribute), 140	(-1	<b></b>
AFL. automation. shared. Diffraction Labeler), 106		eController AFL.automation.loading	(class g.DigitalOutPressure	in Controller),
$\verb other_ttl  (AFL. automation. shared. Server Discovery. Asymptotic Asymptotic and Server Discovery. Asymptotic and Server Discove$	cServiceIn	ifd		
attribute), 124		Controller	(class	in
${\tt other\_ttl} (AFL. automation. shared. Server Discovery. Servattribute), 140$		AFL.automation.loading		
D		eController	(class	in
P		AFL.automation.loading		Controller),
package_cmd() (AFL.automation.loading.UltimusVPressumethod), 78, 79, 179	<i>reControl</i> Pressure	ler.UltimusVPressureCo eControllerAsPump	ntroller (class	in
parsed_addresses() (AFL.automation.shared.ServerDis method), 125		41, 43, 170	g.PressureController.	AsPump),
parsed_addresses() (AFL.automation.shared.ServerDis method), 142		(111 E.automation.shared	l.DataLabelerWidget	t.DataLabelerWidget
narsed scoped addresses()		method), 86, 93, 181		
(AFL.automation.shared.ServerDiscovery.AsyncS	erviceinjo	tton_callback()		**** 1
method), 126		(III L.amomanon.snared	l.DatasetWidget.Date	isetWidget
<pre>parsed_scoped_addresses()</pre>		method), 96, 101, 183		
(AFL.automation.shared.ServerDiscovery.Service	enfov_but	tton_callback()	1.D.M 1.1.1	D:#
method), 142		(III L.antomanon.snaret	l.DiffractionLabeler	DiffractionLabeler
<pre>parsePortName()</pre>		method), 104, 112, 185	1 C A	
AFL. automation. loading. Chemyx Syringe Pump),		y (AFL.automation.share	a.serverDiscovery.A	syncserviceinjo
20, 25, 165		attribute), 124	od SarvarDisaavam. S	anvicaInfo
pause() (AFL.automation.APIServer.Client.Client method), 6, 8, 159		y (AFL.automation.share attribute), 140		
pausePump() (AFL.automation.loading.ChemyxSyringePumethod) 21 25 166	mp.Chemy	signal () (AFL automo xConnection method), 57, 68, 174	ation.loading.Sensor	CallbackThread.Sens

method), 21, 25, 166

```
process_signal() (AFL.automation.loading.SensorCallbackAlly&adSibust&Bbce3KAlHCButomation.loading.SensorPollingThread.Sen
                                                method), 60, 69, 176
                                                                                                                                                                                                                                                                                                                                                       method), 71, 73, 177
process_signal()(AFL.automation.loading.SensorCallbRedClaeDateStopLoadCBv1
                                                                                                                                                                                                                                                                                                                                                      AFL.automation.EpicsADLiveProcess.ReduceDaemon),
                                                method), 63, 69, 175
process_signal() (AFL.automation.loading.SensorCallbackThread\Stop\D6ad\OBv2
                                                method), 66, 69, 176
                                                                                                                                                                                                                                                                                                     register_service() (AFL.automation.shared.ServerDiscovery.Zerocont
properties (AFL. automation. shared. Server Discovery. Async Service Impethod), 146
                                                                                                                                                                                                                                                                                                      reload() (AFL.automation.shared.widgetui.Markdown
                                                 attribute), 126
properties (AFL.automation.shared.ServerDiscovery.ServiceInfo method), 156
                                                                                                                                                                                                                                                                                                      {\tt remove()}\ (AFL. automation. shared. Mutable Queue. Mutable Queue
                                                 attribute), 142
PROTECTED_SAMPLE_KEYS
                                                                                                                                                                                                                                                                                                                                                       method), 113, 114, 186
                                                  (AFL.automation.APIServer.QueueDaemon.DataTremboxee_all_service_listeners()
                                                attribute), 11
                                                                                                                                                                                                                                                                                                                                                       (AFL.automation.shared.ServerDiscovery.Zeroconf
PROTECTED_SYSTEM_KEYS
                                                                                                                                                                                                                                                                                                                                                       method), 146
                                                 (AFL.automation.APIServer.QueueDaemon.DataTirenslossen_item() (AFL.automation.APIServer.Client.Client
                                                 attribute), 11
                                                                                                                                                                                                                                                                                                                                                       method), 6, 9, 159
put() (AFL.automation.shared.MutableQueue.MutableQuememove_listener() (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                method), 113, 114, 186
                                                                                                                                                                                                                                                                                                                                                       method), 148
                                                                                                                                                                                                                                                                                                     remove_service()(AFL.automation.shared.ServerDiscovery.AsyncZeroc
 O
                                                                                                                                                                                                                                                                                                                                                       method), 130
\verb"qsize()" (AFL. automation. shared. Mutable Queue. Mutable Queue. Mutable Queue. Service\_listener()
                                                                                                                                                                                                                                                                                                                                                        (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                 method), 113, 114, 186
                                                                                                                                                                                                                                                                                                                                                       method), 146
query_driver() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                                                                     remove_vertical_lines()
                                                method), 5, 8, 159
{\tt query\_scheduler} (AFL. automation. shared. Server Discovery. Async Server Bitter Bitter at the Server Discovery and 
                                                                                                                                                                                                                                                                                                                                                       method), 84, 93, 182
                                                 attribute), 122
{\tt query\_scheduler} (AFL. automation. shared. Server Discover {\tt P.SEONG} every {\tt Server} all {\tt lines()} in {\tt shared} and {\tt shared} all {\tt lines()} all 
                                                                                                                                                                                                                                                                                                                                                        (AFL. automation. shared. Diffraction Labeler. Diffraction Labeler Vince the Computation of the Computatio
                                                 attribute), 136
                                                                                                                                                                                                                                                                                                                                                       method), 105, 112, 186
queue_state() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                                                                                     Removed (AFL.automation.shared.ServerDiscovery.ServiceStateChange
                                                 method), 6, 8, 159
                                                                                                                                                                                                                                                                                                                                                        attribute), 143
QueueDaemon
                                                                                                                                                              (class
                                                                                                                                                                                                                                                                                 in
                                                                                                                                                                                                                                                                                                      request() (AFL. automation. shared. Server Discovery. Async Service Info
                                                AFL.automation.APIServer.QueueDaemon),
                                                                                                                                                                                                                                                                                                                                                       method), 126
                                                  12, 14, 160
                                                                                                                                                                                                                                                                                                     request() (AFL.automation.shared.ServerDiscovery.ServiceInfo
R
                                                                                                                                                                                                                                                                                                                                                        method), 142
ramp_dispense() (AFL.automation.loading.DigitalOutPressureController_DigitalOutPressureController
                                                                                                                                                                                                                                                                                                                                                        method), 11
                                                 method), 27
{\tt ramp\_dispense()} \ (AFL. automation. loading. Digital OutPressure Controller. Fressure C
                                                                                                                                                                                                                                                                                                                                                        method), 161, 195, 196
                                                  method), 28
{\tt ramp\_dispense()} \ (AFL. automation. loading. Pressure Controller. Pressure Controller. Pressure Controller. AFL. automation. shared. Dataset Widget. Dat
                                                                                                                                                                                                                                                                                                                                                        method), 97, 102, 183
                                                method), 40, 41, 170
{\tt ramp\_dispense()} \ (AFL. automation. loading. Ultimus VP ressure Controller. Pressure Co
                                                                                                                                                                                                                                                                                                                                                        method), 97, 102, 184
method), 77
ramp_dispense() (AFL.automation.loading.UltimusVPressureController.UltimusVPressureController (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.SensorPollingThr
                                                                                                                                                                                                                                                                                                                                                        method), 71, 73, 177
read() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                                                                                                      reset_queue_daemon()
                                                  method), 50, 55, 173
                                                                                                                                                                                                                                                                                                                                                       (AFL.automation.APIServer.Client.Client
read() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                                                                                                                                                       method), 6, 8, 159
                                                method), 52, 55, 174
                                                                                                                                                                                                                                                                                                      reset_sample() (AFL.automation.APIServer.QueueDaemon.DataTrashca
                                                                                     (AFL. automation. loading. Sensor. Sensor
read()
                                                                                                                                                                                                                                                                                                                                                        method), 11
                                                 method), 54, 173
\verb"read()" (AFL. automation. loading. Sensor Polling Thread. Sensor Polling Thread (AFL. automation. loading. Chemyx Syringe Pump. Chemyx Control of the Co
                                                                                                                                                                                                                                                                                                                                                        method), 22, 25, 166
                                                method), 71, 73, 177
```

retrieve\_obj() (AFL.automation.APIServer.Client.Client

```
method), 6, 9, 159
                                                                                                                                                                                                                                                                                           AFL.automation.loading.SainSmartRelay),
revert() (AFL.automation.shared.PersistentConfig.PersistentConfig 45, 47, 171
                                       method), 117, 119, 187
                                                                                                                                                                                                                                                   SampleCell (class in AFL.automation.loading.SampleCell),
run() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
                                       method), 14, 15, 160
                                                                                                                                                                                                                                                   selectPort() (AFL.automation.loading.CetoniMultiPosValve.CetoniMult
run() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDetend), 19, 164
                                       method), 162, 197, 199
                                                                                                                                                                                                                                                   selectPort() (AFL. automation. loading. CetoniMultiPosValve. Flow Selected Selection 1.0 (AFL. automation. loading. CetoniMultiPosValve. Flow Selected Selection 1.0 (AFL. automation. loading. CetoniMultiPosValve. Flow Selected Selecte
run() (AFL.automation.loading.Sensor.DummySensor1
                                                                                                                                                                                                                                                                                            method), 19
                                       method), 50, 55, 173
                                                                                                                                                                                                                                                   selectPort() (AFL. automation. loading. Double Vici Multipos Selector. Dou
run() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                                                                                                                                                            method), 29, 31, 167
                                       method), 52, 55, 174
                                                                                                                                                                                                                                                   selectPort() \ (AFL. automation. loading. Double ViciMultipos Selector. Flow of the control of
run() (AFL.automation.loading.SensorCallbackThread.SensorCallbackEhord);d30
                                                                                                                                                                                                                                                   \verb|selectPort(|)| (AFL. automation. loading. Double Vici Multipos Selector. V
                                       method), 57, 68, 175
run() (AFL.automation.loading.SensorCallbackThread.SimpleThresholdthdhl), 31
                                                                                                                                                                                                                                                   {\tt selectPort()}\ (AFL. automation. loading. Flow Selector. Flow Selector)
                                        method), 61
run() (AFL.automation.loading.SensorCallbackThread.StopLoadCBvilethod), 35, 168
                                                                                                                                                                                                                                                   selectPort() (AFL.automation.loading.ViciMultiposSelector.FlowSelector
                                        method), 64
run() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2ethod), 80
                                                                                                                                                                                                                                                   selectPort() (AFL. automation. loading. ViciMultipos Selector. Vi
                                        method), 67
run() (AFL.automation.loading.SensorPollingThread.SensorPollingThreadd), 81, 179
                                       method), 71, 73, 177
                                                                                                                                                                                                                                                   send() (AFL.automation.shared.ServerDiscovery.Zeroconf
run() (AFL.automation.shared.DataLabelerWidget.DataLabelerViewmethod), 148
                                        method), 84, 93, 182
                                                                                                                                                                                                                                                   send\_command() (AFL. automation. loading. Ultimus VP ressure Controller. U
run() (AFL.automation.shared.DataLabelerWidget.DataLabelerWidgetethod), 78, 79, 179
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.ChemyxSyringePump.ChemyxC
                                       method), 86, 93, 181
run() (AFL.automation.shared.DatasetWidget.DatasetWidget
                                                                                                                                                                                                                                                                                           method), 21, 25, 165
                                        method), 97, 102, 184
                                                                                                                                                                                                                                                    sendCommand() (AFL.automation.loading.DoubleViciMultiposSelector.Ser
run() (AFL.automation.shared.DatasetWidget.DatasetWidget_View method), 30
                                       method), 99, 103, 184
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.DoubleViciMultiposSelector.Vic
run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelemethod), 31
                                        method), 104, 112, 185
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.DummyPump.SerialDevice
run() (AFL.automation.shared.DiffractionLabeler.DiffractionLabelerMixthod), 33
                                       method), 106, 113, 186
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.NE1kSyringePump.SerialDevice
run() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                                                                                                                                                            method), 38
                                        method), 132, 149, 188
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.PressureControllerAsPump.Ser.
run() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                                                                                                                                                                                                                                                                                           method), 43
                                       method), 138
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.SainSmartRelay.SainSmartRela
RunThread (class in AFL.automation.shared.ServerDiscovery),
                                                                                                                                                                                                                                                                                            method), 46
                                         131, 148, 188
                                                                                                                                                                                                                                                   sendCommand() (AFL. automation. loading. SainSmartRelay. SerialDevice
                                                                                                                                                                                                                                                                                           method), 47
S
                                                                                                                                                                                                                                                   sendCommand() (AFL.automation.loading.SerialDevice.SerialDevice
                                                                                                                                                                                                                                                                                            method), 74, 178
sa_aio_discover_server_by_name()
                                        (AFL. automation. API Server. Client. Server Discove \textbf{\$} \underline{\textbf{e}} \textbf{ndCommand()} \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Command()) \ (AFL. automation. Loading. ViciMultipos Selector. Serial Development Comman
                                                                                                                                                                                                                                                                                            method), 80
                                        class method), 7
                                                                                                                                                                                                                                                   sendCommand() (AFL. automation. loading. ViciMultipos Selector. V
sa_aio_discover_server_by_name()
                                        (AFL.automation.shared.ServerDiscovery.ServerDiscovery method), 81
                                                                                                                                                                                                                                                   Sensor (class in AFL.automation.loading.Sensor), 54,
                                        class method), 134, 149, 189
sa_discover_server_by_name()
                                        (AFL. automation. APIS erver. Client. Server Discove \ref{eq:server} ensor Callback Thread
                                                                                                                                                                                                                                                                                                                                                                                                             (class
                                                                                                                                                                                                                                                                                           AFL.automation.loading.SensorCallbackThread),
                                       class method), 7
                                                                                                                                                                                                                                                                                             56, 68, 174
sa_discover_server_by_name()
                                        (AFL. automation. shared. Server Discovery. Server Discovery Polling Thread Server Discover Discovery Polling Thread Server 
                                                                                                                                                                                                                                                                                                                                                                                                           (class
                                                                                                                                                                                                                                                                                           AFL.automation.loading.SensorPollingThread),
                                        class method), 134, 149, 189
                                                                                                                                                                                                                                                                                             70, 73, 176
SainSmartRelay
                                                                                                                                          (class
                                                                                                                                                                                                                                 in
```

SerialCommsException,	74, 150, 151, 190		143
SerialDevice	(class	in	${\tt set\_config()}\ (AFL. automation. APIS erver. Client. Client$
AFL.automation.l	loading.DoubleViciMul	tiposSele	
30			set_driver_object()
SerialDevice	(class	in	(AFL.automation.APIServer.Client.Client
AFL. $automation$ . $l$	loading. Dummy Pump),		method), 6, 9, 159
33			set_object()(AFL.automation.APIServer.Client.Client
SerialDevice	(class	in	method), 6, 9, 159
AFL. $automation$ . $l$	loading.NE1kSyringePı	ump),	$\verb set_output()  (AFL. automation. shared. Data Labeler Widget. Ordinal Encode and the property of the proper$
38			method), 92
SerialDevice	(class		$\verb set_output()  (AFL. automation. shared. Diffraction Labeler. Ordinal Encode and the property of the proper$
AFL. $automation$ . $l$	loading. Pressure Control	ollerAsPı	
42			$\verb set_P()  (AFL. automation. loading. Digital OutPressure Controller. Digital OutPressure Co$
SerialDevice	(class	in	method), 27, 28, 166
AFL.automation.l	loading.SainSmartRela	y),	$\verb set_P()  (AFL. automation. loading. Ultimus VP ressure Controller. Ultimus VP ressure Con$
46			method), 78, 79, 179
SerialDevice	(class	in	$\mathtt{set\_params}() \ (AFL. automation. shared. Data Labeler Widget. Ordinal Encoder States and State$
AFL.automation.l	loading.SerialDevice),	74,	method), 92
177			$\mathtt{set\_params}()$ (AFL. automation. shared. Diffraction Labeler. Ordinal Encodes and States and St
SerialDevice	(class	in	method), 111
AFL.automation.l	loading.ViciMultiposSe	elector),	<pre>set_queue_mode() (AFL.automation.EpicsADLiveProcess.Client.Client</pre>
80			method), 161, 195
serialize()	(in	module	<pre>set_server_if_missing()</pre>
AFL.automation.s	shared.serialization),	151,	(AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
190			method), 126
server (AFL.automation.si	hared.ServerDiscovery.	.AsyncSe	rsateInferver_if_missing()
attribute), 124	·	•	(AFL.automation.shared.ServerDiscovery.ServiceInfo
server (AFL.automation.si	hared.ServerDiscovery.	.ServiceI	
attribute), 140	·		setAllChannelsOff()
server_cmd() (AFL.auton	nation.APIServer.Clien	t.Client	(AFL.automation.loading.SainSmartRelay.SainSmartRelay
method), 5, 8, 158			method), 46, 47, 171
		overy.Asy	v <b>rseSeCharahef</b> bs () (AFL.automation.loading.MultiChannelRelay.MultiChann
attribute), 124		, ,	method), 35, 168
	tion.shared.ServerDisco	overv.Ser	${f nsect Militar}$ ${f annels}$ () (AFL. automation. loading. Sain Smart Relay. MultiChannel R
attribute), 140		-	method), 45
ServerDiscovery	(class	in	setChannels() (AFL.automation.loading.SainSmartRelay.SainSmartRela
	APIServer.Client), 6		method), 46, 47, 171
ServerDiscovery		in	setDaemon() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
_	shared.ServerDiscovery		method), 14
133, 149, 189	·	, , ,	setDaemon() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.Redu
service_state_changed			method), 198
_		v.AsvncS	SesotiDelemons() (AFL.automation.loading.Sensor.DummySensor1
attribute), 122		<i>yy</i>	method), 51
service_state_changed			setDaemon() (AFL.automation.loading.Sensor.DummySensor2
_		v.Service	eBrowser method), 54
attribute), 138		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	setDaemon() (AFL.automation.loading.SensorCallbackThread.SensorCall
ServiceBrowser	(class	in	method), 59
	shared.ServerDiscovery		setDaemon() (AFL.automation.loading.SensorCallbackThread.SimpleThre
134		. , ,	method), 61
ServiceInfo	(class	in	setDaemon() (AFL.automation.loading.SensorCallbackThread.StopLoadC
	shared.ServerDiscovery		method), 64
138		. , ,	setDaemon() (AFL.automation.loading.SensorCallbackThread.StopLoadC

in

method), 67

 $\verb|setDaemon()| (AFL. automation. loading. Sensor Polling Thread. S$ 

(class

AFL. automation. shared. Server Discovery),

 ${\tt ServiceStateChange}$ 

```
method), 73
                                                                                                                                                            method), 39
setDaemon() (AFL.automation.shared.ServerDiscovery.Rus&hRattle() (AFL.automation.loading.PressureControllerAsPump.Pressure
                     method), 133
                                                                                                                                                            method), 42, 44, 171
setDaemon() (AFL.automation.shared.ServerDiscovery.SersetRaton(set(AFL.automation.loading.PressureControllerAsPump.SyringeF
                     method), 138
                                                                                                                                                            method), 43
setdefault() (AFL.automation.APIServer.QueueDaemons@atRaTbaQ)ctAFL.automation.loading.SyringePump.SyringePump
                                                                                                                                                            method), 75, 178
                     method), 12
setdefault() (AFL.automation.shared.DatasetWidget.defaultäime() (AFL.automation.loading.ChemyxSyringePump.ChemyxConnec
                                                                                                                                                            method), 22, 25, 166
                      method), 100
setdefault() (AFL.automation.shared.PersistentConfig.MsatibleMayDinAFL.automation.loading.ChemyxSyringePump.ChemyxConn
                      method), 116
                                                                                                                                                            method), 22, 25, 166
setdefault() (AFL.automation.shared.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.PersistentConfig.Pers
                     method), 118
                                                                                                                                                            method), 12
setDelay() (AFL.automation.loading.ChemyxSyringePump.ChemyxCon.loading.ChemyxSyringePump.ChemyxCon.
                      method), 22, 25, 166
                                                                                                                                                            method), 22, 25, 166
setDiameter() (AFL.automation.loading.ChemyxSyringePsimplGTahresChorldGBion
                                                                                                                                                                                                                      (class
                                                                                                                                                            AFL.automation.loading.SensorCallbackThread),
                     method), 22, 25, 166
setLevel() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
                                                                                                                                      split_vars() (AFL.automation.shared.DatasetWidget.DatasetWidget_Ma
                      method), 23, 25, 165
setName() (AFL.automation.APIServer.QueueDaemon.QueueDaemomethod), 98, 102, 184
                     method), 14
                                                                                                                                      sqrt() (in module AFL.automation.shared.DataLabelerWidget),
setName() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon
                                                                                                                                      sqrt() (in module AFL.automation.shared.DiffractionLabeler),
                     method), 198
setName() (AFL.automation.loading.Sensor.DummySensor1
                     method), 51
                                                                                                                                      start() (AFL.automation.APIServer.QueueDaemon.QueueDaemon
setName() (AFL.automation.loading.Sensor.DummySensor2
                                                                                                                                                            method), 14
                                                                                                                                      start() (AFL. automation. Epics ADLive Process. Reduce Daemon. 
                      method), 54
setName() (AFL.automation.loading.SensorCallbackThread.SensorGaklbackThread
                                                                                                                                      start() (AFL.automation.loading.Sensor.DummySensor1
                     method), 59
setName() (AFL.automation.loading.SensorCallbackThread.SimpleThreatblood)(CB
                      method), 61
                                                                                                                                      start() (AFL.automation.loading.Sensor.DummySensor2
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadi@Brod), 54
                     method), 64
                                                                                                                                      start() (AFL.automation.loading.SensorCallbackThread.SensorCallback
setName() (AFL.automation.loading.SensorCallbackThread.StopLoadleTBv2I), 59
                     method), 67
                                                                                                                                      start() (AFL.automation.loading.SensorCallbackThread.SimpleThreshold
setName() (AFL.automation.loading.SensorPollingThread.SensorPollingTild); and
                     method), 73
                                                                                                                                      start() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv1
setName() (AFL.automation.shared.ServerDiscovery.RunThread
                                                                                                                                                            method), 64
                      method), 133
                                                                                                                                      start() (AFL.automation.loading.SensorCallbackThread.StopLoadCBv2
setName() (AFL.automation.shared.ServerDiscovery.ServiceBrowsemethod), 67
                                                                                                                                      start() (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.SensorPollingT
                     method), 138
setRate() (AFL.automation.loading.ChemyxSyringePump.ChemyxCovertleati)yi73
                                                                                                                                      start() (AFL.automation.shared.ServerDiscovery.RunThread
                     method), 22, 25, 166
setRate() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyminlgoll)unip3
                     method), 23, 25, 165
                                                                                                                                      start() (AFL.automation.shared.ServerDiscovery.ServiceBrowser
setRate() (AFL.automation.loading.ChemyxSyringePump.SyringePumphod), 138
                      method), 24
                                                                                                                                      start() (AFL.automation.shared.ServerDiscovery.Zeroconf
setRate() (AFL.automation.loading.DummyPump.DummyPump
                                                                                                                                                            method), 146
                      method), 32, 34, 168
                                                                                                                                      {\tt started}\,(AFL. automation. shared. Server Discovery. Zero conf
setRate() (AFL.automation.loading.DummyPump.SyringePump
                                                                                                                                                            property), 146
                                                                                                                                      startPump() (AFL.automation.loading.ChemyxSyringePump.ChemyxCon.
                     method), 34
setRate() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePethnopl), 21, 25, 166
                      method), 37, 39, 169
                                                                                                                                      stop() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
```

setRate() (AFL.automation.loading.NE1kSyringePump.SyringePumpethod), 23, 24, 165

```
stop() (AFL.automation.loading.ChemyxSyringePump.SyringePumpmethod), 13, 15, 160
                                                                                                                                                                                                                                                                                                                                          terminate() (AFL.automation.EpicsADLiveProcess.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceDaemon.ReduceD
                                                      method), 24
stop() (AFL.automation.loading.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPressureController.DigitalOutPres
                                                                                                                                                                                                                                                                                                                                         terminate()(AFL.automation.loading.Sensor.DummySensor1
                                                      method), 27
stop() (AFL.automation.loading.DigitalOutPressureController.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.PressureCo
                                                                                                                                                                                                                                                                                                                                          terminate()(AFL.automation.loading.Sensor.DummySensor2
                                                     method), 28
stop() (AFL.automation.loading.DummyPump.DummyPump
                                                                                                                                                                                                                                                                                                                                                                                                method), 52, 55, 174
                                                        method), 32, 34, 168
                                                                                                                                                                                                                                                                                                                                           terminate() (AFL.automation.loading.SensorCallbackThread.SensorCall
stop() (AFL.automation.loading.DummyPump.SyringePump
                                                                                                                                                                                                                                                                                                                                                                                                 method), 57, 68, 174
                                                                                                                                                                                                                                                                                                                                          {\tt terminate()} \ (AFL. automation. loading. Sensor Callback Thread. Simple Thr
                                                      method), 34
stop() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePumpthod), 61
                                                      method), 37, 39, 169
                                                                                                                                                                                                                                                                                                                                          terminate() (AFL.automation.loading.SensorCallbackThread.StopLoadC
stop() (AFL.automation.loading.NE1kSyringePump.SyringePump method), 64
                                                                                                                                                                                                                                                                                                                                          terminate() (AFL.automation.loading.SensorCallbackThread.StopLoadC
                                                      method), 39
stop() (AFL.automation.loading.PressureController.PressureControlleethod), 67
                                                        method), 41, 170
                                                                                                                                                                                                                                                                                                                                          terminate() (AFL. automation. loading. Sensor Polling Thread. Sen
stop() (AFL.automation.loading.PressureControllerAsPump.PressureCitation)
                                                      method), 42, 44, 170
                                                                                                                                                                                                                                                                                                                                         ternary_click_callback()
stop() (AFL:automation.loading.PressureControllerAsPump.Syringe(Afritage automation.shared.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabeler.DiffractionLabele
                                                     method), 43
                                                                                                                                                                                                                                                                                                                                                                                                method), 104, 112, 185
stop() (AFL.automation.loading.SyringePump.SyringePumpest() (in module AFL.automation), 157, 193
                                                     method), 75, 178
                                                                                                                                                                                                                                                                                                                                         {\tt text} (AFL. automation. shared. Server Discovery. Async Service Info
stop() (AFL.automation.loading.UltimusVPressureController.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.PressureCont
                                                                                                                                                                                                                                                                                                                                         {\tt text} (AFL. automation. shared. Server Discovery. Service Info
                                                      method), 77
stop() (AFL.automation.loading.UltimusVPressureController.UltimusVPhase)ureController
                                                                                                                                                                                                                                                                                                                                         \verb|timed_dispense()| (AFL. automation. loading. Digital Out Pressure Controlloading. Digital Out Pressure Controlloading.
                                                      method), 79
stopLoad() (AFL.automation.loading.SensorCallbackThread.LoaderCathordi),pication
                                                     method), 56, 70, 176
                                                                                                                                                                                                                                                                                                                                          \verb|timed_dispense()| (AFL. automation. loading. Digital Out Pressure Controlloading. Digital Out Pressure Controlloading.
StopLoadCBv1
                                                                                                                                                                                                                                                                                                                                                                                                method), 27
                                                                                                                                                                                    (class
                                                                                                                                                                                                                                                                                                                  in
                                                     62, 68, 175
                                                                                                                                                                                                                                                                                                                                                                                                 method), 40, 41, 170
StopLoadCBv2
                                                                                                                                                                                    (class
                                                                                                                                                                                                                                                                                                                                    timed_dispense() (AFL.automation.loading.UltimusVPressureControlle
                                                     AFL.automation.loading.SensorCallbackThread),
                                                                                                                                                                                                                                                                                                                                                                                                 method), 77
                                                      65, 69, 175
                                                                                                                                                                                                                                                                                                                                          timed_dispense() (AFL.automation.loading.UltimusVPressureControlle
 stopPump() (AFL.automation.loading.ChemyxSyringePump.ChemyxGuetheed)ioH
                                                     method), 21, 25, 166
                                                                                                                                                                                                                                                                                                                                          toggleChannels() (AFL. automation. loading. MultiChannelRelay. Mult
 SyringePump
                                                                                                                                                                                 (class
                                                                                                                                                                                                                                                                                                                                                                                                 method), 35, 36, 168
                                                     AFL.automation.loading.ChemyxSyringePump), toggleChannels() (AFL.automation.loading.SainSmartRelay.MultiChannels()
                                                                                                                                                                                                                                                                                                                                                                                                 method), 45
SyringePump
                                                                                                                                                                                                                                                                                                                                       toggleChannels() (AFL.automation.loading.SainSmartRelay.SainSmartI
                                                                                                                                                                                 (class
                                                     AFL.automation.loading.DummyPump),
                                                                                                                                                                                                                                                                                                                                                                                                method), 46, 47, 172
                                                      33
                                                                                                                                                                                                                                                                                                                                          toJSON() (AFL.automation.shared.PersistentConfig.PersistentConfig
                                                                                                                                                                                                                                                                                                                                                                                                 method), 117, 119, 187
SyringePump
                                                                                                                                                                                 (class
                                                                                                                                                                                                                                                                                                                  in
                                                     AFL.automation.loading.NE1kSyringePump),
                                                                                                                                                                                                                                                                                                                                          transfer() (AFL.automation.EpicsADLiveProcess.Client.Client
                                                                                                                                                                                                                                                                                                                                                                                                 method), 161, 195
SyringePump
                                                                                                                                                                                                                                                                                                                                   transform() (AFL.automation.shared.DataLabelerWidget.OrdinalEncode
                                                                                                                                                                                 (class
                                                                                                                                                                                                                                                                                                                   in
                                                     AFL.automation.loading.PressureControllerAsPump),
                                                                                                                                                                                                                                                                                                                                                                                                 method), 90
                                                                                                                                                                                                                                                                                                                                          transform() (AFL.automation.shared.DiffractionLabeler.OrdinalEncoder
SyringePump
                                                                                                                                                                                                                                                                                                                 in
                                                                                                                                                                                                                                                                                                                                                                                                 method), 110
                                                                                                                                                                                 (class
                                                     AFL.automation.loading.SyringePump),
                                                                                                                                                                                                                                                                                                           75,
                                                                                                                                                                                                                                                                                                                                         transmit() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
                                                        178
                                                                                                                                                                                                                                                                                                                                                                                                 method), 11
                                                                                                                                                                                                                                                                                                                                          tubing
                                                                                                                                                                                                                                                                                                                                                                                                      (AFL.automation.loading.Tubing.Tubing
                                                                                                                                                                                                                                                                                                                                                                                                 tribute), 76, 178
terminate() (AFL.automation.APIServer.QueueDaemon.QubiaDaknloos in AFL.automation.loading.Tubing), 76,
```

```
178
                                                                                                                                                                                                     update_plot() (AFL.automation.shared.DiffractionLabeler.DiffractionLa
type (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
                                                                                                                                                                                                                                      method), 104, 112, 185
                                attribute), 124
                                                                                                                                                                                                     update_plot() (AFL.automation.shared.DiffractionLabeler.DiffractionLa
type (AFL.automation.shared.ServerDiscovery.ServiceInfo
                                                                                                                                                                                                                                      method), 105, 112, 186
                                 attribute), 140
                                                                                                                                                                                                     update_plots() (AFL.automation.shared.DatasetWidget.DatasetWidget
types (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowsemethod), 96, 102, 183
                                                                                                                                                                                                     update_record() (AFL.automation.shared.ServerDiscovery.AsyncService
                                 attribute), 122
 types (AFL.automation.shared.ServerDiscovery.ServiceBrowser
                                                                                                                                                                                                                                      method), 122
                                 attribute), 136
                                                                                                                                                                                                     update_record() (AFL.automation.shared.ServerDiscovery.AsyncService
                                                                                                                                                                                                                                      method), 126
U
                                                                                                                                                                                                     update_record() (AFL.automation.shared.ServerDiscovery.ServiceBrown
                                                                                                                                                                                                                                      method), 138
UltimusVPressureController
                                                                                                                                          (class
                                                                                                                                                                                       in
                                AFL. automation. loading. \textit{UltimusVPressureControlpda}, \texttt{te\_record()} \ (AFL. automation. shared. Server Discovery. ServiceInfoliation Discovery. Servic
                                                                                                                                                                                                                                      method), 143
                                  78, 79, 179
                                                                                                                                                                                                     update_sample_dim()
UnpicklingError, 151
                                                                                                                                                                                                                                       (AFL.automation.shared.DatasetWidget.DatasetWidget
unqueued_base() (AFL.automation.APIServer.Client.Client
                                                                                                                                                                                                                                      method), 97, 102, 183
                                method), 5, 8, 158
                                                                                                                                                                                                     update_scattering_plot()
 unregister_all_services()
                                                                                                                                                                                                                                      (AFL.automation.shared.DatasetWidget.DatasetWidget
                                 (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                                                      method), 96, 101, 183
                                method), 148
                                                                                                                                                                                                     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
unregister_service()
                                                                                                                                                                                                                                      method), 99, 103, 184
                                  (AFL.automation.shared.ServerDiscovery.Zeroconf
                                                                                                                                                                                                     update_service()(AFL.automation.shared.ServerDiscovery.AsyncZeroc
                                method), 147
update() (AFL.automation.APIServer.QueueDaemon.DataTrashcan method), 130
                                                                                                                                                                                                     update_service() (AFL.automation.shared.ServerDiscovery.Zeroconf
                                 method), 12
                                                                                                                                                                                                                                      method), 147
update() (AFL.automation.shared.DatasetWidget.defaultdict
                                                                                                                                                                                                     update_status() (AFL.automation.loading.SensorCallbackThread.Sensor
                                method), 100
update() (AFL.automation.shared.PersistentConfig.MutableMappingmethod), 57, 68, 174
                                                                                                                                                                                                     update_status()(AFL.automation.loading.SensorCallbackThread.Simpl
                                 method), 116
update() (AFL.automation.shared.PersistentConfig.PersistentConfig method), 61
                                                                                                                                                                                                     update_status()(AFL.automation.loading.SensorCallbackThread.StopL
                                 method), 117, 119, 187
update_colors() (AFL.automation.shared.DatasetWidget.DatasetWingthpd), 64
                                                                                                                                                                                                     update_status() (AFL.automation.loading.SensorCallbackThread.StopL
                                method), 96, 102, 183
                                                                                                                                                                                                                                      method), 67
update_colorscale()
                                 (AFL. automation. shared. Dataset Widget. Dataset Winglate; \verb|detarmary_colors()| \\
                                                                                                                                                                                                                                      (AFL. automation. shared. Diffraction Labeler. Diffraction Labeler Vin Control of Cont
                                method), 99, 103, 184
                                                                                                                                                                                                                                      method), 106, 113, 186
update_composition_colors()
                                 (AFL. automation. shared. Data Labeler Widget. Data \cite{Labeler Widget}. Data \cit
                                                                                                                                                                                                                                       attribute), 143
                                method), 84, 93, 182
                                                                                                                                                                                                     usbrelay (in module AFL.automation.loading.SainSmartRelay),
update_composition_plot()
                                  (AFL.automation.shared.DatasetWidget.DatasetWidget
                                method), 96, 101, 183
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasetWidget
                                 method), 97, 102, 183
                                                                                                                                                                                                     V40nly (AFL.automation.shared.ServerDiscovery.IPVersion
update_dropdowns() (AFL.automation.shared.DatasetWidget.DatasettWidget), View
                                method), 99, 103, 184
                                                                                                                                                                                                     V6Only\ (AFL. automation. shared. Server Discovery. IPV ersion
update_extract_coords()
                                                                                                                                                                                                                                      attribute), 130
                                 (AFL.automation.shared.DatasetWidget.DatasetWidgetes() (AFL.automation.APIServer.QueueDaemon.DataTrashcan
                                method), 97, 102, 183
                                                                                                                                                                                                                                      method), 12
update_plot() (AFL.automation.shared.DataLabelerWidgea.DurasLubelletViuromation.shared.DatasetWidget.defaultdict
                                                                                                                                                                                                                                      method), 100
                                 method), 84, 93, 182
update_plot() (AFL.automation.shared.DataLabelerWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleEWidgea.DataLabeleE
                                method), 86, 93, 181
                                                                                                                                                                                                                                      method), 116
```

```
values() (AFL.automation.shared.PersistentConfig.PersistentConfig
         method), 118
ViciMultiposSelector
        AFL.automation.loading.DoubleViciMultiposSelector),
ViciMultiposSelector
                                  (class
                                                  in
        AFL.automation.loading.ViciMultiposSelector),
         80, 81, 179
               (AFL.automation.loading.Tubing.Tubing
volume()
        method), 76, 178
W
wait()
              (AFL.automation.APIServer.Client.Client
         method), 5, 8, 158
wait_dosage_finished()
         (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
        method), 23, 24, 165
weight (AFL.automation.shared.ServerDiscovery.AsyncServiceInfo
         attribute), 124
weight (AFL. automation. shared. Server Discovery. Service Info
        attribute), 140
withdraw() (AFL.automation.loading.ChemyxSyringePump.ChemyxSyringePump
        method), 23, 24, 165
withdraw() (AFL.automation.loading.ChemyxSyringePump.SyringePump
        method), 24
withdraw() (AFL.automation.loading.DummyPump.DummyPump
        method), 32, 34, 168
withdraw() (AFL.automation.loading.DummyPump.SyringePump
         method), 34
withdraw() (AFL.automation.loading.NE1kSyringePump.NE1kSyringePump
        method), 37, 39, 169
withdraw() (AFL.automation.loading.NE1kSyringePump.SyringePump
         method), 39
withdraw() (AFL.automation.loading.PressureControllerAsPump.PressureControllerAsPump
        method), 42, 44, 171
withdraw() (AFL.automation.loading.PressureControllerAsPump.SyringePump
         method), 43
withdraw() (AFL.automation.loading.SyringePump.SyringePump
        method), 75, 178
Ζ
zc (AFL.automation.shared.ServerDiscovery.AsyncServiceBrowser
         attribute), 122
zc (AFL.automation.shared.ServerDiscovery.ServiceBrowser
        attribute), 136
Zeroconf (class in AFL.automation.shared.ServerDiscovery),
         143
```