HABApp Documentation

Release beta

spacemanspiff2007

May 26, 2021

USER DOCUMENTATION

1 Installation & Usage **1**
- 1.1 Virtual environment . . . 1
- 1.2 Docker . . . 4
- 1.3 Upgrading to a newer version . . . 4
- 1.4 HABApp arguments . . . 4

2 About HABApp **7**
- 2.1 About . . . 7
- 2.2 HABApp architecture . . . 8
- 2.3 HABApp folder structure . . . 9
- 2.4 Integration with openHAB . . . 9
- 2.5 Integration with MQTT . . . 10

3 Configuration **11**
- 3.1 Configuration contents . . . 11

4 Getting Started **13**
- 4.1 First rule . . . 13
- 4.2 A more generic rule . . . 14
- 4.3 Interacting with items . . . 14
- 4.4 Watch items for events . . . 15
- 4.5 Trigger an event when an item is constant . . . 16

5 Logging **19**
- 5.1 Configuration . . . 19

6 Rule **21**
- 6.1 Interacting with items . . . 21
- 6.2 Events . . . 22
- 6.3 Scheduler . . . 23
- 6.4 Running external tools . . . 27
- 6.5 How to properly use rules from other rule files . . . 27
- 6.6 All available functions . . . 28

7 Parameters **31**
- 7.1 Parameters . . . 31
- 7.2 Validation . . . 32
- 7.3 Create rules from Parameters . . . 33
- 7.4 Parameter classes . . . 33

8 HABApp **35**

8.1 Items . . . 35
8.2 Events . . . 43

9 openHAB **47**
9.1 Additional configuration . . . 47
9.2 Interaction with a openHAB . . . 48
9.3 Openhab item types . . . 51
9.4 Openhab event types . . . 76
9.5 Textual thing configuration . . . 83
9.6 Example openHAB rules . . . 92

10 MQTT **97**
10.1 Interaction with the MQTT broker . . . 97
10.2 Rule Interface . . . 97
10.3 Mqtt item types . . . 98
10.4 Mqtt event types . . . 102
10.5 Example MQTT rule . . . 103

11 Advanced Usage **105**
11.1 HABApp Topics . . . 105
11.2 File properties . . . 106
11.3 Invoking OpenHAB actions . . . 106
11.4 Mocking OpenHAB items and events for tests . . . 108

12 asyncio **109**
12.1 async http . . . 109

13 util - helpers and utilities **113**
13.1 Functions . . . 113
13.2 CounterItem . . . 115
13.3 Statistics . . . 116
13.4 MultiModeItem . . . 117

14 Additional rule examples **125**
14.1 Using the scheduler . . . 125
14.2 Mirror openHAB events to a MQTT Broker . . . 126
14.3 Trigger an event when an item is constant . . . 126
14.4 Turn something off after movement . . . 127
14.5 Process Errors in Rules . . . 128

15 Tips & Tricks **129**
15.1 yml files . . . 129
15.2 openHAB . . . 129

16 Class reference **131**
16.1 Watches . . . 131
16.2 Scheduler . . . 132

17 Indices and tables **139**

Python Module Index **141**

Index **143**

CHAPTER

ONE

INSTALLATION & USAGE

1.1 Virtual environment

1.1.1 Installation

Hint:

With openhabian the complete installation can be performed through the openhabian-config tool (option 2B). HABApp will be installed into `/opt/habapp`, so it is the same as the installation described here.

Hint: On Windows use the `python` command instead of `python3`

1. Navigate to the folder where the virtual environment shall be created (e.g.):

```
cd /opt
```

2. Create virtual environment (this will create a new folder "habapp"):

```
python3 -m venv habapp
```

3. Go into folder of virtual environment:

```
cd habapp
```

4. Activate the virtual environment

   Linux:

```
source bin/activate
```

   Windows:

```
Scripts\activate
```

5. Upgrade pip:

```
python3 -m pip install --upgrade pip
```

6. Install HABApp:

```
python3 -m pip install habapp
```

7. Run HABAp:

```
habapp --config PATH_TO_CONFIGURATION_FOLDER
```

A good configuration folder for HABApp would be your openhab configuration folder (e.g. `/opt/openhab/conf/habapp` or `/etc/openhab/habapp`) because this is where your other configuration folders are located (e.g. the items and sitemaps folder). Just make sure to manually create the folder `habapp` before the start.

Hint: After the installation take a look how to configure HABApp. A default configuration will be created on the first start.

1.1.2 Upgrading

1. Stop HABApp
2. Activate the virtual environment

   Navigate to the folder where HABApp is installed:

   ```
   cd /opt/habapp
   ```

   Activate the virtual environment

   Linux:

   ```
   source bin/activate
   ```

   Windows:

   ```
   Scripts\activate
   ```

3. Run the following command in your activated virtual environment:

```
python3 -m pip install --upgrade habapp
```

4. Start HABApp
5. Observe the logs for errors in case there were changes

1.1.3 Autostart after reboot

Check where habapp is installed

```
which habapp
```

To automatically start HABApp from the virtual environment after a reboot call:

```
nano /etc/systemd/system/habapp.service
```

and copy paste the following contents. If the user which is running openhab is not "openhab" replace accordingly. If your installation is not done in "/opt/habapp/bin" replace accordingly as well:

```
[Unit]
Description=HABApp
Documentation=https://habapp.readthedocs.io
After=network-online.target

[Service]
Type=simple
User=openhab
Group=openhab
UMask=002
ExecStart=/opt/habapp/bin/habapp -c PATH_TO_CONFIGURATION_FOLDER

[Install]
WantedBy=multi-user.target
```

Press Ctrl + x to save.

Now execute the following commands to enable autostart:

```
sudo systemctl --system daemon-reload
sudo systemctl enable habapp.service
```

It is now possible to start, stop, restart and check the status of HABApp with:

```
sudo systemctl start habapp.service
sudo systemctl stop habapp.service
sudo systemctl restart habapp.service
sudo systemctl status habapp.service
```

1.1.4 Error message while installing ujson

Under windows the installation of ujson may throw the following error but the download link is not working. Several working alternatives can be found here.

```
Running setup.py install for ujson ... error
  ERROR: Complete output from command 'C:\Users\User\Desktop\HABapp\habapp\Scripts\
↪python.exe' -u -c 'import setuptools, tokenize;__file__='"'"'C:\\Users\\User\\AppData\\
↪Local\\Temp\\pip-install-4y0tobjp\\ujson\\setup.py'"'"';f=getattr(tokenize, '"'"'open'"
↪'"', open)(__file__);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();
↪exec(compile(code, __file__, '"'"'exec'"'"'))' install --record 'C:\Users\User\AppData\
↪Local\Temp\pip-record-6t2yo712\install-record.txt' --single-version-externally-managed␣
↪--compile --install-headers 'C:\Users\User\Desktop\HABapp\habapp\include\site\python3.
↪7\ujson':
  ERROR: Warning: 'classifiers' should be a list, got type 'filter'
  running install
  running build
  running build_ext
  building 'ujson' extension
  error: Microsoft Visual C++ 14.0 is required. Get it with "Microsoft Visual C++ Build␣
↪Tools": https://visualstudio.microsoft.com/downloads/
  ----------------------------------------
```

1.1.5 Error message while installing ruamel.yaml

```
_ruamel_yaml.c:4:10: fatal error: Python.h: No such file or directory
```

Run the follwing command to fix it:

```
sudo apt install python3-dev
```

1.2 Docker

1.2.1 Installation

Installation through docker is also available:

```
docker pull spacemanspiff2007/habapp
```

To have the proper timestamps in the logs set the `TZ` environment variable of the container accordingly (e.g. `TZ=Europe/Berlin`).

1.2.2 Updating docker on Synology

To update your HABApp docker within Synology NAS, you just have to do the following:

On the Synology NAS just select "Download" with tag "latest" to download the new image. It will overwrite the old one on the NAS. Then stop the container. After selecting "Action" -> "Clear" on the HABapp container, the container is there, but without any content. After starting the container again, everything should immediately work again.

1.3 Upgrading to a newer version

It is recommended to upgrade the installation on another machine. Configure your production instance in the configuration and set the `listen_only` switch(es) in the configuration to `True`. Observe the logs for any errors. This way if there were any breaking changes rules can easily be fixed before problems occur on the running installation.

1.4 HABApp arguments

Execute habapp with "-h" to view possible command line arguments

```
habapp -h
```

```
usage: -c [-h] [-c CONFIG] [-wos WAIT_OS_UPTIME] [-b]

Start HABApp

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Path to configuration folder (where the config.yml is
```

(continues on next page)

(continued from previous page)

```
                      located)
-wos WAIT_OS_UPTIME, --wait_os_uptime WAIT_OS_UPTIME
                      Waits for the specified os uptime before starting
                      HABApp
-b, --benchmark       Do a Benchmark based on the current config
```

CHAPTER

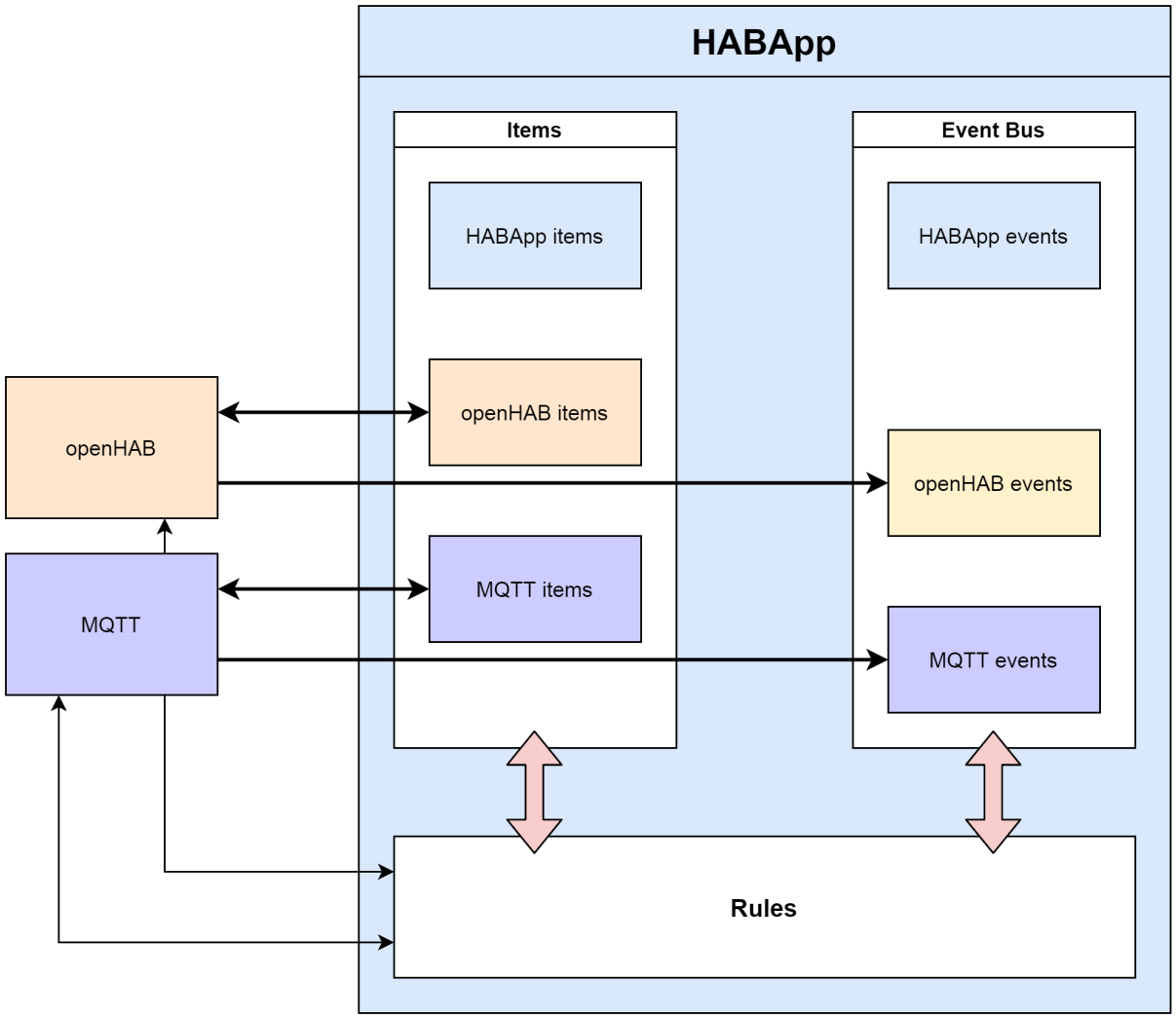
TWO

ABOUT HABAPP

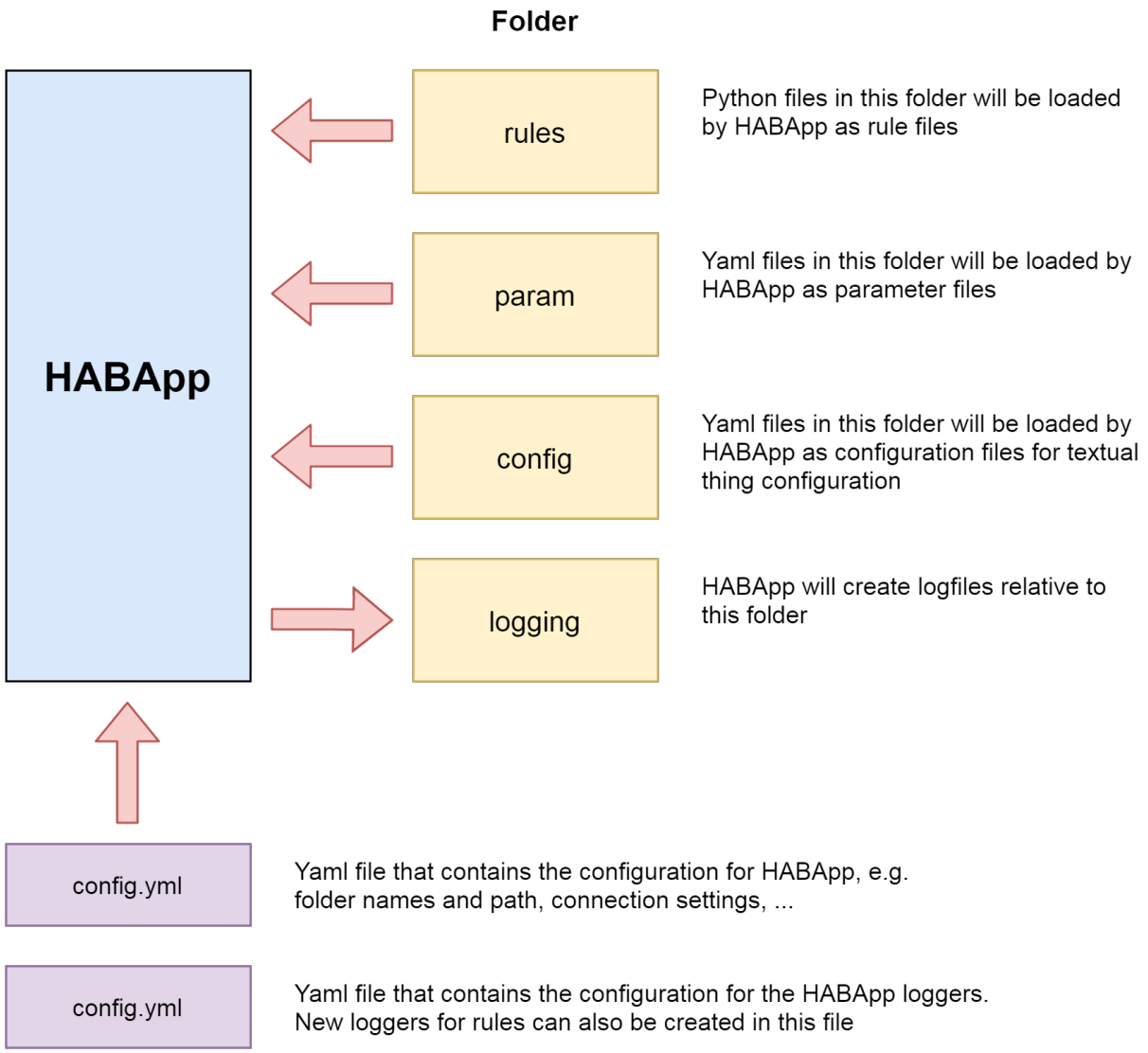
2.1 About

HABApp is a Python rule engine for home automation. It has local items, an event bus and can integrate external systems, e.g. openHAB and MQTT. Rules can listen to events from the event bus. These events are generated by HABApp or by the external systems. Additionally there is a scheduler available that makes time based triggering very easy.

2.2 HABApp architecture



2.3 HABApp folder structure



2.4 Integration with openHAB

HABApp connects to the openhab event stream and automatically updates the local openhab items when an item in openhab changes. These item values are cached, so accessing and working with items in rules is very fast. The events from openhab are also mirrored to the internal event bus which means that triggering on these events is also possible.

When HABApp connects to openhab for the first time it will load all items/things from the openhab instance and create local items. The name of the local openhab items is equal to the name in openhab.

Posting updates, sending commands or any other openhab interface call will issue a corresponding REST-API call to change openhab.

2.5 Integration with MQTT

HABApp subscribes to the defined mqtt topics. For every MQTT message with the `retain` flag HABApp will automatically create an *MqttItem* so these values can be accessed later. The name of the created item is the the mqtt topic. All other messages will **not** automatically create an item but still create an event on the event bus.

MqttItems created by rules will automatically be updated with the latest value once a message is received. These item values are cached, so accessing and working with items in rules is very fast.

CHAPTER

THREE

CONFIGURATION

Configuration is done through `config.yml` The parent folder of the file can be specified with `-c PATH` or `--config PATH`. If nothing is specified the file `config.yml` is searched in the subdirectory *HABApp* in

- the current working directory
- the venv directory
- the user home

If the config does not yet exist in the folder a blank configuration will be created

3.1 Configuration contents

```yaml
directories:
    logging: log    # If the filename for the logfile in logging.yml is not absolute it␣
↪will be placed in this directory
    rules: rules    # All *.py files in this folder (and subfolders) will be loaded.␣
↪Load order will be alphabetical by path.
    param: param    # Optional, this is the folder where the parameter files will be␣
↪created and loaded from
    config: config  # Folder from which configuration files for openhab will be loaded
    lib: lib        # Custom modules, libraries and files can be placed there.
                    # (!) Attention (!):
                    # Don't create rule instances in files inside the lib folder! It will␣
↪lead to strange behaviour.

location:           # Specify the location where your HABApp instance is running
  latitude: 0.0     # The value is used to calculate the Sunrise/Sunset etc accordingly
  longitude: 0.0
  elevation: 0.0

openhab:
    ping:
        enabled: true        # If enabled the configured item will show how long it␣
↪takes to send an update from HABApp
                             # and get the updated value back in milliseconds
        item: 'HABApp_Ping'  # Name of the Numberitem that will show the ping
        interval: 10         # Seconds between two pings

    connection:
```

(continues on next page)

(continued from previous page)

```
        host: localhost
        port: 8080
        user: ''
        password: ''

    general:
        listen_only: False  # If True  HABApp will not change any value on the openhab␣
↪instance.
                            # Useful for testing rules from another machine.
        wait_for_openhab: True   # If True HABApp will wait for items from the openHAB␣
↪instance
                                 # before loading any rules on startup


mqtt:
    connection:
        client_id: HABApp
        host: ''
        port: 8883
        user: ''
        password: ''
        tls: true
        tls_insecure: false  # do not check certificate

    subscribe:          # Changes to Subscribe get picked up without restarting HABApp
        qos: 0          # Default QoS for subscribing
        topics:
        - '#'           # Subscribe to this topic
        - 0             # QoS for previous topic, can be omitted

    publish:
        qos: 0           # Default QoS when publishing values
        retain: false    # Default retain flag when publishing values

    general:
        listen_only: False # If True  HABApp will not publish any value to the broker.
                           # Useful for testing rules from another machine.
```

CHAPTER

FOUR

GETTING STARTED

It is really recommended to use a python IDE, for example PyCharm. The IDE can provide auto complete and static checks which will help you write error free rules and vastly speed up your developement.

First start HABApp and keep it running. It will automatically load and update all rules which are created or changed in the configured `rules` directory. Loading and unloading of rules can be observed in the HABApp logfile.

It is recommended to use HABApp from the console for these examples so the print output can be observed.

4.1 First rule

Rules are written as classes that inherit from *HABApp.Rule*. Once the class gets instantiated the will run as rules in the HABApp rule engine. So lets write a small rule which prints something.

```
import HABApp

# Rules are classes that inherit from HABApp.Rule
class MyFirstRule(HABApp.Rule):
    def __init__(self):
        super().__init__()

        # Use run.at to schedule things directly after instantiation,
        # don't do blocking things in __init__
        self.run.soon(self.say_something)

    def say_something(self):
        print('That was easy!')

# Rules
MyFirstRule()
```

```
That was easy!
```

4.2 A more generic rule

It is also possible to instantiate the rules with parameters. This often comes in handy if there is some logic that shall be applied to different items.

```
import HABApp

class MyFirstRule(HABApp.Rule):
    def __init__(self, my_parameter):
        super().__init__()
        self.param = my_parameter

        self.run.soon(self.say_something)

    def say_something(self):
        print(f'Param {self.param}')

# This is normal python code, so you can create Rule instances as you like
for i in range(2):
    MyFirstRule(i)
for t in ['Text 1', 'Text 2']:
    MyFirstRule(t)
```

```
Param 0
Param 1
Param Text 1
Param Text 2
```

4.3 Interacting with items

HABApp uses an internal item registry to store both openhab items and locally created items (only visible within HABApp). Upon start-up HABApp retrieves a list of openhab items and adds them to the internal registry. Rules and HABApp derived libraries may add additional local items which can be used to share states across rules and/or files.

An item is created and added to the item registry through the corresponding class factory method

```
from HABApp.core.items import Item

# This will create an item in the local (HABApp) item registry
item = Item.get_create_item("an-item-name", "a value")
```

Posting values from the item will automatically create the events on the event bus. This example will create an item in HABApp (locally) and post some updates to it. To access items from openhab use the correct openhab item type (see *the openhab item description*).

```
import HABApp
from HABApp.core.items import Item
```

(continues on next page)

(continued from previous page)

```
class MyFirstRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        # Get the item or create it if it does not exist
        self.my_item = Item.get_create_item('Item_Name')

        self.run.soon(self.say_something)

    def say_something(self):
        # Post updates to the item through the internal event bus
        self.my_item.post_value('Test')
        self.my_item.post_value('Change')

        # The item value can be used in comparisons through this shortcut ...
        if self.my_item == 'Change':
            print('Item value is "Change"')
        # ... which is the same as this:
        if self.my_item.value == 'Change':
            print('Item.value is "Change"')


MyFirstRule()
```

Output

```
[HABApp.EventBus]     INFO |            Item_Name: <ValueUpdateEvent name: Item_Name,␣
↪value: Test>
[HABApp.EventBus]     INFO |            Item_Name: <ValueChangeEvent name: Item_Name,␣
↪value: Test, old_value: None>
[HABApp.EventBus]     INFO |            Item_Name: <ValueUpdateEvent name: Item_Name,␣
↪value: Change>
[HABApp.EventBus]     INFO |            Item_Name: <ValueChangeEvent name: Item_Name,␣
↪value: Change, old_value: Test>
Item value is "Change"
Item.value is "Change"
```

4.4 Watch items for events

It is possible to watch items for changes or updates.

```
import HABApp
from HABApp.core.items import Item
from HABApp.core.events import ValueUpdateEvent, ValueChangeEvent

class MyFirstRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        # Get the item or create it if it does not exist
        self.my_item = Item.get_create_item('Item_Name')
```

(continues on next page)

(continued from previous page)

```
        # Run this function whenever the item receives an ValueUpdateEvent
        self.listen_event(self.my_item, self.item_updated, ValueUpdateEvent)

        # Run this function whenever the item receives an ValueChangeEvent
        self.listen_event(self.my_item, self.item_changed, ValueChangeEvent)

        # If you already have an item you can use the more convenient method of the item
        self.my_item.listen_event(self.item_changed, ValueChangeEvent)

    # the function has 1 argument which is the event
    def item_changed(self, event: ValueChangeEvent):
        print(f'{event.name} changed from "{event.old_value}" to "{event.value}"')
        print(f'Last change of {self.my_item.name}: {self.my_item.last_change}')

    def item_updated(self, event: ValueUpdateEvent):
        print(f'{event.name} updated value: "{event.value}"')
        print(f'Last update of {self.my_item.name}: {self.my_item.last_update}')

MyFirstRule()
```

```
Item_Name updated value: "Changed value"
Last update of Item_Name: 2021-05-26T11:34:59.551646
Item_Name changed from "Some value" to "Changed value"
Last change of Item_Name: 2021-05-26T11:34:59.551646
Item_Name changed from "Some value" to "Changed value"
Last change of Item_Name: 2021-05-26T11:34:59.551646
```

4.5 Trigger an event when an item is constant

```
import HABApp
from HABApp.core.items import Item
from HABApp.core.events import ItemNoChangeEvent

class MyFirstRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        # Get the item or create it if it does not exist
        self.my_item = Item.get_create_item('Item_Name')

        # This will create an event if the item is 10 secs constant
        watcher = self.my_item.watch_change(10)

        # this will automatically listen to the correct event
        watcher.listen_event(self.item_constant)

        # To listen to all ItemNoChangeEvent/ItemNoUpdateEvent independent of the␣
↪timeout time use
        # self.listen_event(self.my_item, self.item_constant, watcher.EVENT)
```

(continues on next page)

(continued from previous page)

```
    def item_constant(self, event: ItemNoChangeEvent):
        print(f'{event}')

MyFirstRule()
```

```
<ItemNoChangeEvent name: Item_Name, seconds: 10>
```

CHAPTER

FIVE

LOGGING

5.1 Configuration

5.1.1 Example usage

The logging library is the standard python library.

```
import logging

import HABApp

log = logging.getLogger('MyRule')


class MyLoggingRule(HABApp.Rule):

    def __init__(self):
        super().__init__()

        # different levels are available
        log.debug('Debug Message')
        log.info('Info Message')
        log.warning('Warning Message')
        log.error('Error Message')


MyLoggingRule()
```

5.1.2 Example configuration

Configuration of logging is done through `logging.yml`. During the first start a default configuration will be created. It is recommended to extend the default configuration.

The complete description of the file format can be found here, but the format should be pretty straight forward.

Hint:

It is highly recommended to use absolute paths as file names
e.g.: `/HABApp/logs/my_logfile.log` or `c:\HABApp\logs\my_logfile.log`

```yaml
# describes the output format
formatters:
  HABApp_format:
    format: '[%(asctime)s] [%(name)25s] %(levelname)8s | %(message)s'

# describes the available file handlers
handlers:
  HABApp_default:
    class: HABApp.core.lib.handler.MidnightRotatingFileHandler
    filename: 'HABApp.log'
    maxBytes: 10_000_000
    backupCount: 3

    formatter: HABApp_format  # use the specified formatter (see above)
    level: DEBUG

  MyRuleHandler:
    class: HABApp.core.lib.handler.MidnightRotatingFileHandler
    filename: 'c:\HABApp\Logs\MyRule.log'    # absolute filename is recommended
    maxBytes: 10_000_000
    backupCount: 3

    formatter: HABApp_format  # use the specified formatter (see above)
    level: DEBUG


# List all available loggers
loggers:
  HABApp:
    level: DEBUG
    handlers:
      - HABApp_default  # This logger does log with the default handler
    propagate: False

  MyRule:   # Name of the logger, see example usage
    level: DEBUG
    handlers:
      - MyRuleHandler  # This logger uses the MyRuleHandler
    propagate: False
```

5.1.3 Custom log levels

It is possible to add custom log levels or rename existing levels. This is possible via the optional `levels` entry in the logging configuration file.

```yaml
levels:
  WARNING: WARN  # Changes WARNING to WARN
  5: TRACE       # Adds a new loglevel "TRACE" with value 5

formatters:
  HABApp_format:
...
```

CHAPTER

SIX

RULE

6.1 Interacting with items

Items are like variables. They have a name and a value (which can be anything). Items from openhab use the item name from openhab and get created when HABApp successfully connects to openhab or when the openhab configuration changes. Items from MQTT use the topic as item name and get created as soon as a message gets processed.

Some item types provide convenience functions, so it is advised to always set the correct item type.

The preferred way to get and create items is through the class factories *get_item* and *get_create_item* since this ensures the proper item class and provides type hints when using an IDE! Example:

```
from HABApp.core.items import Item
my_item = Item.get_create_item('MyItem', initial_value=5)   # This will create the item␣
↪if it does not exist
my_item = Item.get_item('MyItem')                           # This will raise an␣
↪exception if the item is not found
print(my_item)
```

If an item value gets set there will be a `ValueUpdateEvent` on the event bus. If it changes there will be additionally a `ValueChangeEvent`, too.

It is possible to check the item value by comparing it

```
from HABApp.core.items import Item
my_item = Item.get_item('MyItem')

# this works
if my_item == 5:
    pass    # do something

# and is the same as this
if my_item.value == 5:
    pass    # do something
```

An overview over the item types can be found on *the HABApp item section*, *the openhab item section* and the *the mqtt item section*

6.2 Events

It is possible to listen to events through the *listen_event()* function. The passed function will be called as soon as an event occurs and the event will pe passed as an argument into the function.

There is the possibility to reduce the function calls to a certain event type with an additional parameter (typically `ValueUpdateEvent` or `ValueChangeEvent`).

An overview over the events can be found on *the HABApp event section*, *the openhab event section* and the *the mqtt event section* Example

```
from HABApp import Rule
from HABApp.core.events import ValueChangeEvent, ValueUpdateEvent
from HABApp.core.items import Item

class MyRule(Rule):
    def __init__(self):
        super().__init__()
        self.listen_event('MyOpenhabItem', self.on_change, ValueChangeEvent)    # will␣
↪trigger only on ValueChangeEvent
        self.listen_event('My/MQTT/Topic', self.on_update, ValueUpdateEvent)    # will␣
↪trigger only on ValueUpdateEvent

        # If you already have an item you can and should use the more convenient method␣
↪of the item
        # to listen to the item events
        my_item = Item.get_item('MyItem')
        my_item.listen_event(self.on_change, ValueUpdateEvent)

    def on_change(self, event: ValueChangeEvent):
        assert isinstance(event, ValueChangeEvent), type(event)

    def on_update(self, event: ValueUpdateEvent):
        assert isinstance(event, ValueUpdateEvent), type(event)

MyRule()
```

Additionally there is the possibility to filter not only on the event type but on the event values, too. This can be achieved by passing an **instance** of EventFilter as event type. There are convenience Filters (e.g. *ValueUpdateEventFilter* and *ValueChangeEventFilter*) for the most used event types that provide type hints.

class HABApp.core.events.**EventFilter**(*event_type*, ***kwargs*)

class HABApp.core.events.**ValueUpdateEventFilter**(*value*)

class HABApp.core.events.**ValueChangeEventFilter**(*value=<_MissingType.MISSING: <object object>>*, *old_value=<_MissingType.MISSING: <object object>>*)

Example

```
from HABApp import Rule
from HABApp.core.events import EventFilter, ValueUpdateEventFilter, ValueUpdateEvent
from HABApp.core.items import Item

class MyRule(Rule):
```

(continues on next page)

(continued from previous page)

```
    def __init__(self):
        super().__init__()
        my_item = Item.get_item('MyItem')

        # This will only call the callback for ValueUpdateEvents where the value==my_
↪value
        my_item.listen_event(self.on_val_my_value, ValueUpdateEventFilter(value='my_value
↪'))

        # This is the same as above but with the generic filter
        my_item.listen_event(self.on_val_my_value, EventFilter(ValueUpdateEvent, value=
↪'my_value'))

    def on_val_my_value(self, event: ValueUpdateEvent):
        assert isinstance(event, ValueUpdateEvent), type(event)

MyRule()
```

6.3 Scheduler

With the scheduler it is easy to call functions in the future or periodically. Do not use *time.sleep* but rather *self.run.at*. Another very useful function is *self.run.countdown* as it can simplify many rules!

Function	Description
soon()	Run the callback as soon as possible (typically in the next second).
at()	Run the callback in x seconds or at a specified time.
countdown()	Run a function after a time has run down
every()	Run a function periodically
every_minute()	Run a function every minute
every_hour()	Run a function every hour
on_every_day()	Run a function at a specific time every day
on_workdays()	Run a function at a specific time on workdays
on_weekends()	Run a function at a specific time on weekends
on_day_of_week()	Run a function at a specific time on specific days of the week
on_sun_dawn()	Run a function on dawn
on_sunrise()	Run a function on sunrise
on_sunset()	Run a function on sunset
on_sun_dusk()	Run a function on dusk

All functions return an instance of ScheduledCallbackBase

class HABApp.rule.habappscheduler.**HABAppScheduler**(*rule*)

at(*time*, *callback*, **args*, ***kwargs*)
Create a a job that will run at a specified time.

Parameters

- **time** (Union[None, datetime, timedelta, time, int]) –
- **callback** – Function which will be called

- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *OneTimeJob*

Returns Created job

countdown(*expire_time*, *callback*, **args*, ***kwargs*)
Run a job a specific time after calling `reset()` of the job. Another subsequent call to `reset()` will start the countdown again.

Parameters

- **expire_time** (Union[timedelta, float, int]) – countdown in seconds or a timedelta obj
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *CountdownJob*

Returns Created job

every(*start_time*, *interval*, *callback*, **args*, ***kwargs*)
Create a job that will run at a specific interval.

Parameters

- **start_time** (Union[None, datetime, timedelta, time, int]) – First execution time
- **interval** (Union[int, float, timedelta]) – Interval how the job is repeated
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *ReoccurringJob*

Returns Created job

on_day_of_week(*time*, *weekdays*, *callback*, **args*, ***kwargs*)
Create a job that will run at a certain time on certain days during the week.

Parameters

- **time** (Union[time, datetime]) – Time when the job will run
- **weekdays** (Union[str, Iterable[Union[str, int]]]) – Day group names (e.g. `'all'`, `'weekend'`, `'workdays'`), an iterable with day names (e.g. `['Mon', 'Fri']`) or an iterable with the isoweekday values (e.g. `[1, 5]`).
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DayOfWeekJob*

Returns Created job

on_every_day(*time*, *callback*, **args*, ***kwargs*)
Create a job that will run at a certain time of day

Parameters

- **time** (Union[time, datetime]) – Time when the job will run
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DayOfWeekJob*

on_sunrise(*callback*, **args*, ***kwargs*)

Create a job that will run on sunrise, requires a location to be set

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *SunriseJob*

Returns Created job

on_sunset(*callback*, **args*, ***kwargs*)

Create a job that will run on sunset, requires a location to be set

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *SunsetJob*

Returns Created job

on_sun_dawn(*callback*, **args*, ***kwargs*)

Create a job that will run on dawn, requires a location to be set

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DawnJob*

Returns Created job

on_sun_dusk(*callback*, **args*, ***kwargs*)

Create a job that will run on dusk, requires a location to be set

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DuskJob*

Returns Created job

soon(*callback*, **args*, ***kwargs*)
Run the callback as soon as possible.

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *OneTimeJob*

every_minute(*callback*, **args*, ***kwargs*)
Picks a random second and runs the callback every minute

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *ReoccurringJob*

every_hour(*callback*, **args*, ***kwargs*)
Picks a random minute and second and run the callback every hour

Parameters

- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *ReoccurringJob*

on_weekends(*time*, *callback*, **args*, ***kwargs*)
Create a job that will run at a certain time on weekends.

Parameters

- **time** (Union[time, datetime]) – Time when the job will run
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DayOfWeekJob*

Returns Created job

on_workdays(*time*, *callback*, **args*, ***kwargs*)
Create a job that will run at a certain time on workdays.

Parameters

- **time** (Union[time, datetime]) – Time when the job will run
- **callback** – Function which will be called
- **args** – Positional arguments that will be passed to the function
- **kwargs** – Keyword arguments that will be passed to the function

Return type *DayOfWeekJob*

Returns Created job

6.4 Running external tools

External tools can be run with the *execute_subprocess()* function. Once the process has finished the callback will be called with an *FinishedProcessInfo* instance as first argument. Example:

```
import HABApp

class MyExecutionRule(HABApp.Rule):

    def __init__(self):
        super().__init__()

        self.execute_subprocess( self.func_when_finished, 'path_to_program', 'arg1',␣
↪capture_output=True)

    def func_when_finished(self, process_info):
        assert isinstance(process_info, HABApp.rule.FinishedProcessInfo)
        print(process_info)

MyExecutionRule()
```

class HABApp.rule.**FinishedProcessInfo**(*returncode*, *stdout*, *stderr*)

Information about the finished process.

Variables

- **returncode** (*int*) – Return code of the process (0: IO, -1: Exception while starting process)
- **stdout** (*str*) – Standard output of the process or None
- **stderr** (*str*) – Error output of the process or None

6.5 How to properly use rules from other rule files

This example shows how to properly get a rule during runtime and execute one of its function. With the proper import and type hint this method provides syntax checks and auto complete.

Rule instances can be accessed by their name (typically the class name). In the `HABApp.log` you can see the name when the rule is loaded. If you want to assign a custom name, you can change the rule name easily by assigning it to `self.rule_name` in `__init__`.

Important: Always look up rule every time, never assign to a class member! The rule might get reloaded and then the class member will still point to the old unloaded instance.

---

rule_a.py:

```
import HABApp
```

(continues on next page)

(continued from previous page)

```
class ClassA(HABApp.Rule):
    ...

    def function_a(self):
      ...

ClassA()
```

rule_b.py:

```
import HABApp
import typing

if typing.TYPE_CHECKING:            # This is only here to allow
    from .rule_a import ClassA      # type hints for the IDE

class ClassB(HABApp.Rule):
    ...

    def function_b(self):

        r = self.get_rule('ClassA')  # type: ClassA
        # The comment "# type: ClassA" will signal the IDE that the value returned from␣
↪the
        # function is an instance of ClassA and thus provide checks and auto complete.

        # this calls the function on the instance
        r.function_a()
```

6.6 All available functions

class HABApp.**Rule**

Variables

- **async_http** – *Async http connections*
- **mqtt** – *MQTT interaction*
- **openhab** – *Openhab interaction*
- **oh** – short alias for openhab openhab

post_event(*name*, *event*)

Post an event to the event bus

Parameters

- **name** – name or item to post event to
- **event** – Event class to be used (must be class instance)

Returns

listen_event(*name*, *callback*, *event_type=<class 'HABApp.core.events.events.AllEvents'>*)

Register an event listener

Parameters

- **name** (Union[*BaseValueItem*, str]) – item or name to listen to. Use None to listen to all events
- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[Type[AllEvents], *EventFilter*, Any]) – Event filter. This is typically *ValueUpdateEvent* or *ValueChangeEvent* which will also trigger on changes/update from openhab or mqtt. Additionally it can be an instance of *EventFilter* which additionally filters on the values of the event. There are also templates for the most common filters, e.g. *ValueUpdateEventFilter* and *ValueChangeEventFilter*

Return type EventBusListener

execute_subprocess(*callback*, *program*, **args*, *capture_output=True*)

Run another program

Parameters

- **callback** – Function which will be called after process has finished. First parameter will be an instance of *FinishedProcessInfo*
- **program** – program or path to program to run
- **args** – Positional arguments that will be passed to the function
- **capture_output** – Capture program output, set to *False* to only capture return code

Returns

register_on_unload(*func*)

Register a function with no parameters which will be called when the rule is unloaded. Use this for custom cleanup functions.

Parameters **func** (Callable[[], Any]) – function which will be called

register_cancel_obj(*obj*)

Add a weakref to an obj which has a cancel function. When the rule gets unloaded the cancel function will be called (if the obj was not already garbage collected)

Parameters **obj** –

CHAPTER

SEVEN

PARAMETERS

7.1 Parameters

Parameters are values which can easily be changed without having to reload the rules. Values will be picked up during runtime as soon as they get edited in the corresponding file. If the file doesn't exist yet it will automatically be generated in the configured *param* folder. Parameters are perfect for boundaries (e.g. if value is below param switch something on). Currently there are is *Parameter* and *DictParameter* available.

```
import HABApp

class MyRuleWithParameters(HABApp.Rule):
    def __init__(self):
        super().__init__()

        # construct parameter once, default_value can be anything
        self.min_value = HABApp.Parameter( 'param_file_testrule', 'min_value', default_
↪value=10)

        # deeper structuring is possible through specifying multiple keys
        self.min_value_nested = HABApp.Parameter(
            'param_file_testrule',
            'Rule A', 'subkey1', 'subkey2',
            default_value=['a', 'b', 'c'] # defaults can also be dicts or lists
        )

        self.listen_event('test_item', self.on_change_event, HABApp.core.events.
↪ValueChangeEvent)

    def on_change_event(self, event):

        # the parameter can be used like a normal variable, comparison works as expected
        if self.min_value < event.value:
            pass

        # The current value can be accessed through the value-property, but don't cache␣
↪it!
        current_value = self.min_value.value


MyRuleWithParameters()
```

Created file:

```
min_value: 10
Rule A:
    subkey1:
        subkey2:
            - a
            - b
            - c
```

Changes in the file will be automatically picked up through *Parameter*.

7.2 Validation

Since parameters used to provide flexible configuration for automation classes they can get quite complex and error prone. Thus it is possible to provide a validator for a file which will check the files for constraints, missing keys etc. when the file is loaded.

HABApp.parameters.**set_file_validator**(*filename*, *validator*, *allow_extra_keys=True*)

Add a validator for the parameter file. If the file is already loaded this will reload the file.

Parameters

- **filename** (str) – filename which shall be validated (without extension)
- **validator** (Any) – Description of file content - see the library voluptuous for examples. Use *None* to remove validator.
- **allow_extra_keys** – Allow additional keys in the file structure

Example

```
import HABApp
import voluptuous

# Validator can even and should be specified before loading rules

# allows a dict e.g. { 'key1': {'key2': '5}}
HABApp.parameters.set_file_validator('file1', {str: {str: int}})

# More complex example with an optional key:
validator = {
    'Test': int,
    'Key': {
        'mandatory_key': str,
        voluptuous.Optional('optional'): int
    }
}
HABApp.parameters.set_file_validator('file1', validator)
```

7.3 Create rules from Parameters

Parameteres are not bound to rule instance and thus work everywhere in the rule file. It is possible to dynamically create rules from the contents of the parameter file.

It's even possible to automatically reload rules if the parameter file has changed: Just add the "reloads on" entry to the file.

Listing 1: my_param.yml

```yaml
key1:
  v: 10
key2:
  v: 12
```

rule

```
import HABApp

class MyRule(HABApp.Rule):
    def __init__(self, k, v):
        super().__init__()

        print(f'{k}: {v}')


cfg = HABApp.DictParameter('my_param')    # this will get the file content
for k, v in cfg.items():
    MyRule(k, v)
```

```
key1: {'v': 10}
key2: {'v': 12}
```

7.4 Parameter classes

class HABApp.parameters.**Parameter**(*filename*, **keys*, *default_value='ToDo'*)

Class to dynamically access parameters which are loaded from file.

Parameters

- **filename** (str) – filename (without extension)
- **keys** – structure in the file
- **default_value** (Any) – default value for the parameter. Is used to create the file and the structure if it does not exist yet. Use None to skip creation of the file structure.

property value: Any

Return the current value. This will do the lookup so make sure to not cache this value, otherwise the parameter might not work as expected.

Return type Any

class HABApp.parameters.**DictParameter**(*filename*, **keys*, *default_value='ToDo'*)

Implements a dict interface

Class to dynamically access parameters which are loaded from file.

Parameters

- **filename** (str) – filename (without extension)
- **keys** – structure in the file
- **default_value** (Any) – default value for the parameter. Is used to create the file and the structure if it does not exist yet. Use None to skip creation of the file structure.

property value: dict

Return the current value. This will do the lookup so make sure to not cache this value, otherwise the parameter might not work as expected.

Return type dict

CHAPTER

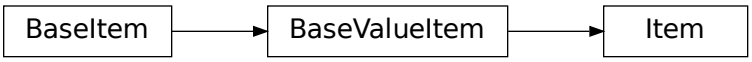
EIGHT

HABAPP

This page describes the HABApp internals

8.1 Items

8.1.1 Item



class HABApp.core.items.**Item**(*name*, *initial_value=None*)

Simple item, used to store values in HABApp

classmethod get_create_item(*name*, *initial_value=None*)

Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

- **name** (str) – item name
- **initial_value** – state the item will have if it gets created

Returns item

classmethod get_item(*name*)

Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_value(*default_value=None*)

Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)

Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_value(*new_value*)

Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)

Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)

Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)

Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

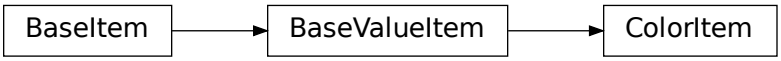
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

8.1.2 ColorItem



class HABApp.core.items.**ColorItem**(*name*, *hue=0.0*, *saturation=0.0*, *brightness=0.0*)

Item for dealing with color related values

classmethod get_item(*name*)

Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_rgb(*max_rgb_value=255*)

Return a rgb equivalent of the color

Parameters **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536

Return type Tuple[int, int, int]

Returns rgb tuple

get_value(*default_value=None*)

Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_off()

Return true if item is off

Return type bool

is_on()

Return true if item is on

Return type bool

listen_event(*callback*, *event_type*)

Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_rgb(*r*, *g*, *b*, *max_rgb_value=255*)
Set a new rgb value and post appropriate events on the HABApp event bus (`ValueUpdateEvent`, `ValueChangeEvent`)

Parameters

- **r** – red value
- **g** – green value
- **b** – blue value
- **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536

Return type *ColorItem*

Returns self

post_value(*hue=0.0*, *saturation=0.0*, *brightness=0.0*)
Set a new value and post appropriate events on the HABApp event bus (`ValueUpdateEvent`, `ValueChangeEvent`)

Parameters

- **hue** – hue (in °)
- **saturation** – saturation (in %)
- **brightness** – brightness (in %)

set_rgb(*r*, *g*, *b*, *max_rgb_value=255*, *ndigits=2*)
Set a rgb value

Parameters

- **r** – red value
- **g** – green value
- **b** – blue value
- **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536
- **ndigits** (`Optional[int]`) – Round the hsb values to the specified digits, None to disable rounding

Return type *ColorItem*

Returns self

set_value(*hue=0.0*, *saturation=0.0*, *brightness=0.0*)
Set the color value

Parameters

- **hue** – hue (in °)
- **saturation** – saturation (in %)
- **brightness** – brightness (in %)

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (`Union[int, float, timedelta]`) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)

Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

8.1.3 AggregationItem

The aggregation item is an item which takes the values of another item in a time period as an input. It then allows to process these values and generate an aggregated output based on it. The item makes implementing time logic like "Has it been dark for the last hour?" or "Was there frost during the last six hours?" really easy. And since it is just like a normal item triggering on changes etc. is possible, too.

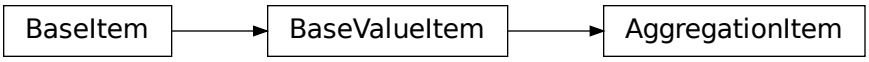
```
from HABApp.core.items import AggregationItem
my_agg = AggregationItem.get_create_item('MyAggregationItem')

# Connect the source item with the aggregation item
my_agg.aggregation_source('MyInputItem')

# Aggregate all changes in the last two hours
my_agg.aggregation_period(2 * 3600)

# Use max as an aggregation function
my_agg.aggregation_func = max
```

The value of `my_agg` in the example will now always be the maximum of `MyInputItem` in the last two hours. It will automatically update and always reflect the latest changes of `MyInputItem`.



class HABApp.core.items.**AggregationItem**(*name*)

aggregation_func(*func*)
Set the function which will be used to aggregate all values. E.g. `min` or `max`

Parameters **func** (`Callable[[Iterable], Any]`) – The function which takes an iterator an returns an aggregated value. Important: the function must be **non blocking**!

Return type *AggregationItem*

aggregation_period(*period*)
Set the period in which the items will be aggregated

Parameters **period** (`Union[float, int, timedelta]`) – period in seconds

Return type *AggregationItem*

aggregation_source(*source*, *only_changes=False*)
Set the source item which changes will be aggregated

Parameters

- **source** (`Union[BaseValueItem, str]`) – name or Item obj
- **only_changes** (`bool`) – if true only value changes instead of value updates will be added

Return type *AggregationItem*

classmethod get_create_item(*name*)
Creates a new AggregationItem in HABApp and returns it or returns the already existing one with the given name

Parameters **name** (`str`) – item name

Returns item

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (`str`) – Name of the item

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type `Any`

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_value(*new_value*)

Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)

Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)

Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)

Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

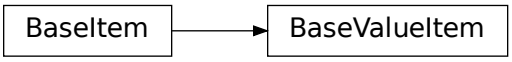
property name: str

Return type str

Returns Name of the item (read only)

8.1.4 BaseValueItem

Base class for items with values. All items that have a value must inherit from *BaseValueItem* May not be instantiated directly.



class HABApp.core.items.**BaseValueItem**(*name*, *initial_value=None*)

Simple item

Variables

- **name** (*str*) – Name of the item (read only)
- **value** – Value of the item, can be anything (read only)
- **last_change** (*datetime.datetime*) – Timestamp of the last time when the item has changed the value (read only)
- **last_update** (*datetime.datetime*) – Timestamp of the last time when the item has updated the value (read only)

classmethod get_item(*name*)

Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_value(*default_value=None*)

Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)

Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_value(*new_value*)

Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

`set_value`(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

`watch_change`(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

`watch_update`(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

8.2 Events

8.2.1 ValueUpdateEvent

This event gets emitted every time a value of an item receives an update

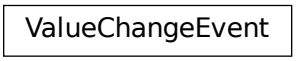
class HABApp.core.events.**ValueUpdateEvent**(*name=None*, *value=None*)

Variables

- **name** (*str*) –
- **value** –

8.2.2 ValueChangeEvent

This event gets emitted every time a value of an item changes



class HABApp.core.events.**ValueChangeEvent**(*name=None*, *value=None*, *old_value=None*)

Variables

- **name** (*str*) –
- **value** –
- **old_value** –

8.2.3 ItemNoUpdateEvent

This event gets emitted when an item is watched for updates and no update has been made in a certain amount of time.

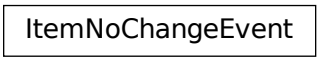
class HABApp.core.events.**ItemNoUpdateEvent**(*name=None*, *seconds=None*)

Variables

- **name** (*str*) –
- **float] seconds** (*Union[int,*) –

8.2.4 ItemNoChangeEvent

This event gets emitted when an item is watched for changes and no change has been made in a certain amount of time.



class HABApp.core.events.**ItemNoChangeEvent**(*name=None*, *seconds=None*)

Variables

- **name** (*str*) –
- **float] seconds** (*Union[int,*) –

CHAPTER

NINE

OPENHAB

9.1 Additional configuration

9.1.1 openHAB 2

For openHAB2 there is no additional configuration needed.

9.1.2 openHAB 3

For optimal performance it is recommended to use Basic Auth (available from openHAB 3.1 M3 on). It can be enabled through GUI or through textual configuration.

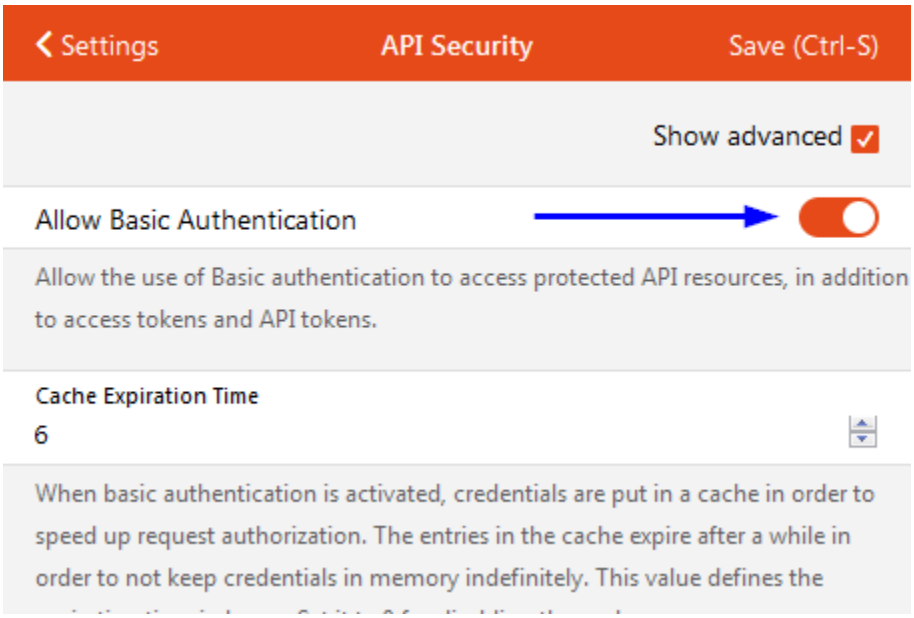
Textual configuration

The settings are in the `runtime.cfg`. Remove the # before the entry to activate it.

```
################ REST API ###################
org.openhab.restauth:allowBasicAuth=true
```

GUI

It can be enabled through the gui in `settings` -> `API Security` -> `Allow Basic Authentication`.



9.2 Interaction with a openHAB

All interaction with the openHAB is done through the `self.oh` or `self.openhab` object in the rule or through an `OpenhabItem`.

9.2.1 Function parameters

`HABApp.openhab.interface.`**`post_update`**(*item_name*, *state*)
Post an update to the item

Parameters

- **`item_name`** (`str`) – item name or item
- **`state`** (`Any`) – new item state

`HABApp.openhab.interface.`**`send_command`**(*item_name*, *command*)
Send the specified command to the item

Parameters

- **`item_name`** (`str`) – item name or item
- **`command`** – command

`HABApp.openhab.interface.`**`get_item`**(*item_name*, *metadata=None*)
Return the complete OpenHAB item definition

Parameters

- **`item_name`** (`str`) – name of the item or item

- **metadata** (Optional[str]) – metadata to include (optional, comma separated or search expression)

Return type OpenhabItemDefinition

Returns

HABApp.openhab.interface.**item_exists**(*item_name*)
Check if an item exists in the OpenHAB item registry

Parameters **item_name** (str) – name

HABApp.openhab.interface.**remove_item**(*item_name*)
Removes an item from the openHAB item registry

Parameters **item_name** (str) – name

HABApp.openhab.interface.**create_item**(*item_type*, *name*, *label=''*, *category=''*, *tags=[]*, *groups=[]*, *group_type=''*, *group_function=''*, *group_function_params=[]*)
Creates a new item in the OpenHAB item registry or updates an existing one

Parameters

- **item_type** (str) – item type
- **name** (str) – item name
- **label** – item label
- **category** – item category
- **tags** (List[str]) – item tags
- **groups** (List[str]) – in which groups is the item
- **group_type** (str) – what kind of group is it
- **group_function** (str) – group state aggregation function
- **group_function_params** (List[str]) – params for group state aggregation

Returns True if item was created/updated

HABApp.openhab.interface.**get_thing**(*thing_name*)
Return the complete OpenHAB thing definition

Parameters **thing_name** (str) – name of the thing or the item

Return type OpenhabThingDefinition

HABApp.openhab.interface.**get_persistence_data**(*item_name*, *persistence*, *start_time*, *end_time*)
Query historical data from the OpenHAB persistence service

Parameters

- **item_name** (str) – name of the persistet item
- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

Return type OpenhabPersistenceData

HABApp.openhab.interface.**remove_metadata**(*item_name*, *namespace*)
Remove metadata from an item

Parameters

- **item_name** (str) – name of the item or item
- **namespace** (str) – namespace

Returns

HABApp.openhab.interface.**set_metadata**(*item_name*, *namespace*, *value*, *config*)
Add/set metadata to an item

Parameters

- **item_name** (str) – name of the item or item
- **namespace** (str) – namespace
- **value** (str) – value
- **config** (dict) – configuration

Returns

HABApp.openhab.interface.**get_channel_link**(*channel_uid*, *item_name*)
returns the ItemChannelLinkDefinition for a link between a (things) channel and an item

Parameters

- **channel_uid** (str) – uid of the (things) channel (usually something like AAAA:BBBBB:CCCCC:DDDD:0#SOME_NAME)
- **item_name** (str) – name of the item

Return type ItemChannelLinkDefinition

Returns an instance of ItemChannelLinkDefinition or None on error

HABApp.openhab.interface.**remove_channel_link**(*channel_uid*, *item_name*)
removes a link between a (things) channel and an item

Parameters

- **channel_uid** (str) – uid of the (thing) channel (usually something like AAAA:BBBBB:CCCCC:DDDD:0#SOME_NAME)
- **item_name** (str) – name of the item

Return type bool

Returns true on successful removal, otherwise false

HABApp.openhab.interface.**channel_link_exists**(*channel_uid*, *item_name*)
check if a things channel is linked to an item

Parameters

- **channel_uid** (str) – uid of the linked channel (usually something like AAAA:BBBBB:CCCCC:DDDD:0#SOME_NAME)
- **item_name** (str) – name of the linked item

Return type bool

Returns true when the link exists, otherwise false

HABApp.openhab.interface.**create_channel_link**(*channel_uid*, *item_name*, *configuration=None*)
creates a link between a (things) channel and an item

Parameters

- **channel_uid** (str) – uid of the (thing) channel (usually something like AAAA:BBBBB:CCCCC:DDDD:0#SOME_NAME)
- **item_name** (str) – name of the item
- **configuration** (Optional[Dict[str, Any]]) – optional configuration for the channel

Return type bool

Returns true on successful creation, otherwise false

9.3 Openhab item types

9.3.1 Description and example

Items that are created from openHAB inherit all from OpenhabItem and provide convenience functions which simplify many things.

Example:

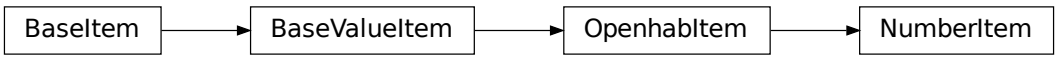
```
from HABApp.openhab.items import ContactItem, SwitchItem

my_contact = ContactItem.get_item('MyContact')
if my_contact.is_open():
    print('Contact is open!')

my_switch = SwitchItem.get_item('MySwitch')
if my_switch.is_on():
    my_switch.off()
```

```
Contact is open!
```

9.3.2 NumberItem



class HABApp.openhab.items.**NumberItem**(*name*, *initial_value=None*)

NumberItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)

Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)

Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

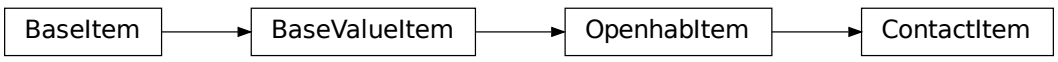
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.3 ContactItem



class HABApp.openhab.items.**ContactItem**(*name*, *initial_value=None*)

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used

- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_closed()
Test value against closed-value

Return type bool

is_open()
Test value against open-value

Return type bool

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

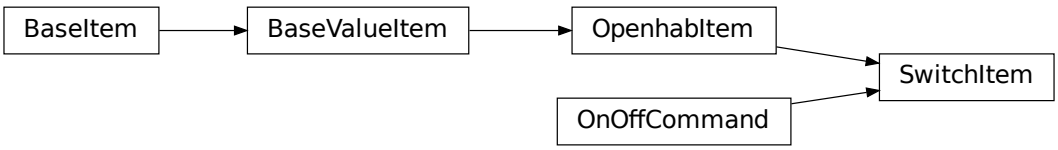
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.4 SwitchItem



class HABApp.openhab.items.**SwitchItem**(*name*, *initial_value=None*)

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_off()
Test value against off-value

Return type bool

is_on()
Test value against on-value

Return type bool

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

off()
Command item off

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

on()
Command item on

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

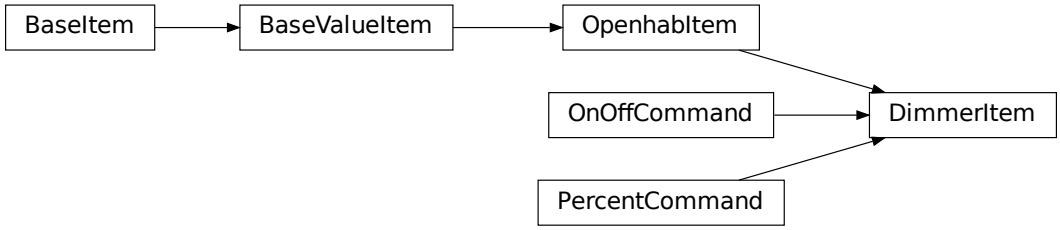
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.5 DimmerItem



class HABApp.openhab.items.**DimmerItem**(*name*, *initial_value=None*)

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_off()
Test value against off-value

Return type bool

is_on()
Test value against on-value

Return type bool

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event

- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

off()
Command item off

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

on()
Command item on

percent(*value*)
Command to value (in percent)

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

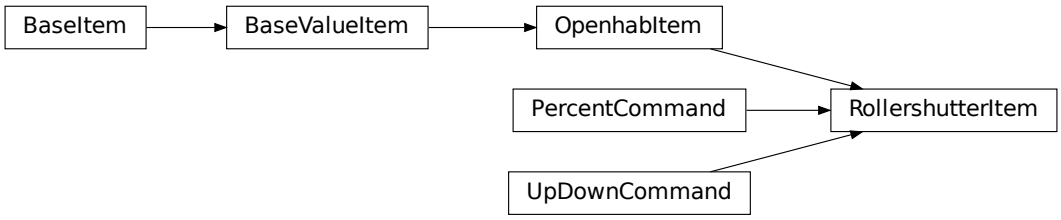
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.6 RollershutterItem



class HABApp.openhab.items.**RollershutterItem**(*name*, *initial_value=None*)

down()
Command down

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_down()
Test value against off-value

Return type bool

is_up()
Test value against on-value

Return type bool

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

percent(*value*)
Command to value (in percent)

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

up()
Command up

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

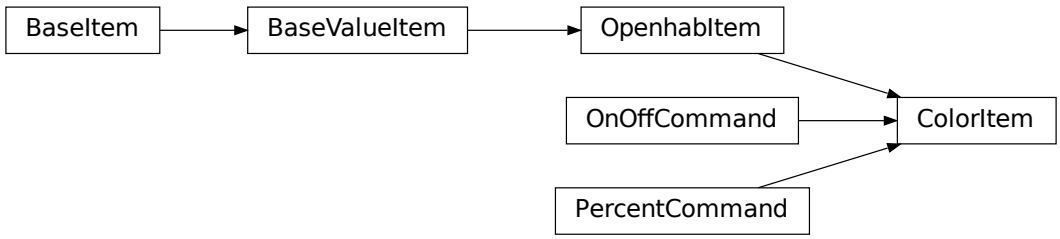
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.7 ColorItem



class HABApp.openhab.items.**ColorItem**(*name*, *h=0.0*, *s=0.0*, *b=0.0*)

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_rgb(*max_rgb_value=255*)
Return a rgb equivalent of the color

Parameters **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536

Return type Tuple[int, int, int]

Returns rgb tuple

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

is_off()
Return true if item is off

Return type bool

is_on()
Return true if item is on

Return type bool

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

off()
Command item off

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

on()
Command item on

percent(*value*)
Command to value (in percent)

post_rgb(*r*, *g*, *b*, *max_rgb_value=255*)
Set a new rgb value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters

- **r** – red value
- **g** – green value
- **b** – blue value
- **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536

Return type *ColorItem*

Returns self

post_value(*hue=0.0*, *saturation=0.0*, *brightness=0.0*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters

- **hue** – hue (in °)
- **saturation** – saturation (in %)
- **brightness** – brightness (in %)

set_rgb(*r*, *g*, *b*, *max_rgb_value=255*, *ndigits=2*)
Set a rgb value

Parameters

- **r** – red value
- **g** – green value
- **b** – blue value
- **max_rgb_value** – the max value for rgb, typically 255 (default) or 65.536
- **ndigits** (Optional[int]) – Round the hsb values to the specified digits, None to disable rounding

Return type *ColorItem*

Returns self

set_value(*hue=0.0*, *saturation=0.0*, *brightness=0.0*)
Set the color value

Parameters

- **hue** – hue (in °)

- **saturation** – saturation (in %)
- **brightness** – brightness (in %)

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

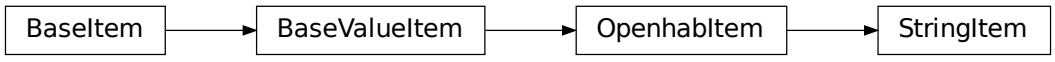
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.8 StringItem



class HABApp.openhab.items.**StringItem**(*name*, *initial_value=None*)
StringItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

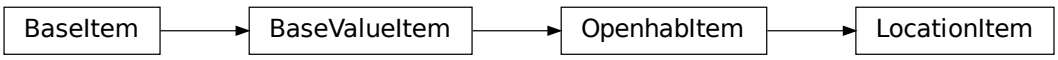
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.9 LocationItem



class HABApp.openhab.items.**LocationItem**(*name*, *initial_value=None*)
LocationItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

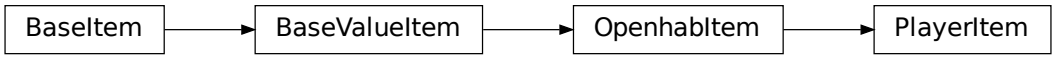
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.10 PlayerItem



class HABApp.openhab.items.**PlayerItem**(*name*, *initial_value=None*)
PlayerItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used

- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

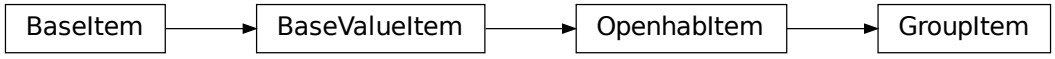
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.11 GroupItem



class HABApp.openhab.items.**GroupItem**(*name*, *initial_value=None*)
GroupItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*value=<_MissingType.MISSING: <object object>>*)
Post an update to the openHAB item

Parameters **value** (Any) – (optional) value to be posted. If not specified the item value will be used.

oh_send_command(*value=<_MissingType.MISSING: <object object>>*)
Send a command to the openHAB item

Parameters **value** (Any) – (optional) value to be sent. If not specified the item value will be used.

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

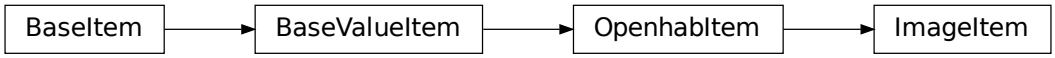
Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.12 ImageItem



class HABApp.openhab.items.**ImageItem**(*name*, *initial_value=None*)
ImageItem which accepts and converts the data types from OpenHAB

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_persistence_data(*persistence=None*, *start_time=None*, *end_time=None*)
Query historical data from the OpenHAB persistence service

Parameters

- **persistence** (Optional[str]) – name of the persistence service (e.g. rrd4j, mapdb). If not set default will be used
- **start_time** (Optional[datetime]) – return only items which are newer than this
- **end_time** (Optional[datetime]) – return only items which are older than this

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

oh_post_update(*data*, *img_type=None*)
Post an update to an openhab image with new image data. Image type is automatically detected, in rare cases when this does not work it can be set manually.

Parameters

- **data** (bytes) – image data
- **img_type** (Optional[str]) – (optional) what kind of image, jpeg or png

oh_send_command(*data*, *img_type=None*)
Send a command to an openhab image with new image data. Image type is automatically detected, in rare cases when this does not work it can be set manually.

Parameters

- **data** (bytes) – image data
- **img_type** (Optional[str]) – (optional) what kind of image, jpeg or png

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

9.3.13 Thing

class HABApp.openhab.items.**Thing**(*name*)
Base class for Things

Variables **status** (*str*) – Status of the thing (e.g. OFFLINE, ONLINE, . . .)

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

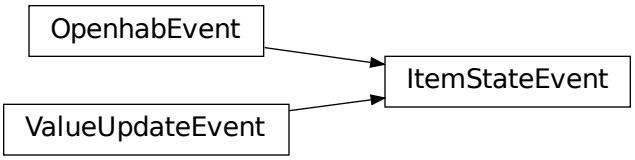
9.4 Openhab event types

Openhab produces various events that are mapped to the internal event bus. On the OpenHab page there is an explanation for the various events.

9.4.1 Item events

ItemStateEvent

Since this event inherits from *ValueUpdateEvent* you can listen to *ValueUpdateEvent* and it will also trigger for *ItemStateEvent*.



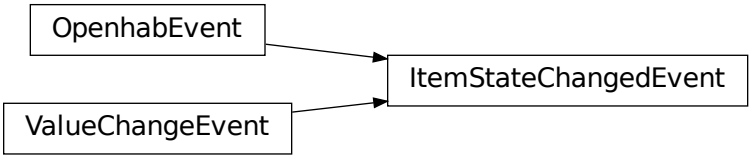
class HABApp.openhab.events.**ItemStateEvent**(*name=''*, *value=None*)

Variables

- **name** (*str*) –
- **value** –

ItemStateChangedEvent

Since this event inherits from *ValueChangeEvent* you can listen to *ValueChangeEvent* and it will also trigger for *ItemStateChangedEvent*.

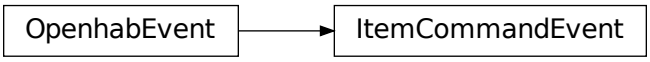


class HABApp.openhab.events.**ItemStateChangedEvent**(*name=''*, *value=None*, *old_value=None*)

Variables

- **name** (*str*) –
- **value** –
- **old_value** –

ItemCommandEvent

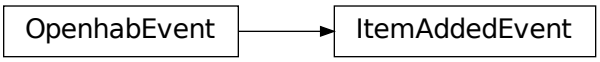


class HABApp.openhab.events.**ItemCommandEvent**(*name=''*, *value=None*)

Variables

- **name** (*str*) –
- **value** –

ItemAddedEvent

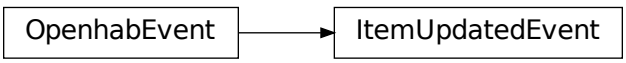


class HABApp.openhab.events.**ItemAddedEvent**(*name=''*, *type=''*)

Variables

- **name** (*str*) –
- **type** (*str*) –

ItemUpdatedEvent

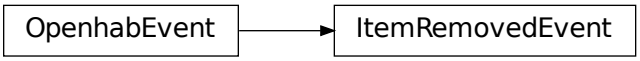


class HABApp.openhab.events.**ItemUpdatedEvent**(*name=''*, *type=''*)

Variables

- **name** (*str*) –
- **type** (*str*) –

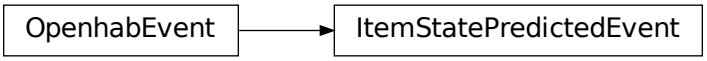
ItemRemovedEvent



class HABApp.openhab.events.**ItemRemovedEvent**(*name=''*)

Variables **name** (*str*) –

ItemStatePredictedEvent



class HABApp.openhab.events.**ItemStatePredictedEvent**(*name=''*, *value=None*)

Variables

- **name** (*str*) –
- **value** –

GroupItemStateChangedEvent

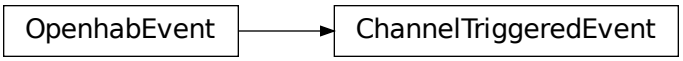
class HABApp.openhab.events.**GroupItemStateChangedEvent**(*name=''*, *item=''*, *value=None*, *old_value=None*)

Variables

- **name** (*str*) –
- **item** (*str*) –
- **value** –
- **old_value** –

9.4.2 Channel events

ChannelTriggeredEvent



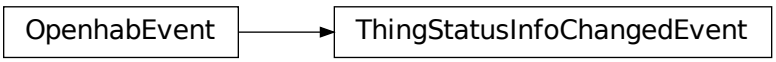
class HABApp.openhab.events.**ChannelTriggeredEvent**(*name=''*, *event=''*, *channel=''*)

Variables

- **name** (*str*) –
- **event** (*str*) –
- **channel** (*str*) –

9.4.3 Thing events

ThingStatusInfoChangedEvent



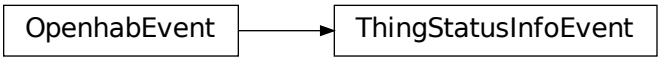
class HABApp.openhab.events.**ThingStatusInfoChangedEvent**(*name=''*, *status=''*, *detail=''*, *old_status=''*, *old_detail=''*)

Variables

- **name** (*str*) –
- **status** (*str*) –
- **detail** (*str*) –

- **old_status** (*str*) –
- **old_detail** (*str*) –

ThingStatusInfoEvent

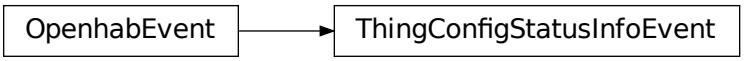


class HABApp.openhab.events.**ThingStatusInfoEvent**(*name=''*, *status=''*, *detail=''*)

Variables

- **name** (*str*) –
- **status** (*str*) –
- **detail** (*str*) –

ThingConfigStatusInfoEvent

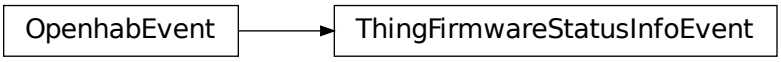


class HABApp.openhab.events.**ThingConfigStatusInfoEvent**(*name=''*, *messages=[{}]*)

Variables

- **name** (*str*) –
- **messages** (*list*) –

ThingFirmwareStatusInfoEvent



class HABApp.openhab.events.**ThingFirmwareStatusInfoEvent**(*name=''*, *status=''*)

Variables

- **name** (*str*) –
- **status** (*str*) –

9.4.4 Event filters

ItemStateEventFilter

class HABApp.openhab.events.**ItemStateEventFilter**(*value*)

ItemStateChangedEventFilter

class HABApp.openhab.events.**ItemStateChangedEventFilter**(*value=<_MissingType.MISSING: <object object>>*, *old_value=<_MissingType.MISSING: <object object>>*)

9.5 Textual thing configuration

9.5.1 Description

HABApp offers a special mechanism to textually define thing configuration parameters and linked items for things which have been added through the gui. This combines the best of both worlds: auto discovery, easy and fast sharing of parameters and items across things.

Configuration is done in the `thing_your_name.yml` file in the `config` folder (see *Configuration*). Every file that starts with `thing_` has the `.yml` ending will be loaded.

The Parameters and items will be checked/set when HABApp connects to openHAB or whenever the corresponding file gets changed.

9.5.2 Principle of operation

All existing things from openHAB can be filtered by different criteria. For each one of these remaining things it is then possible to

- Set thing parameters
- Create items with values taken from the thing fields
- Apply filters to the channels of the thing
  For each matching channel it is possible to create and link items with values taken from the thing and the matching channel values

There is also a test mode which prints out all required information and does not make any changes.

A valid `.items` file will automatically be created next to the `.yml` file containing all created items. It can be used to get a quick overview what items (would) have been created or copied into the items folder.

9.5.3 File Structure

Configuration is done through a .yml file.

Example

The following example will show how to set the Z-Wave Parameters 4, 5, 6 and 8 for a `Philio PST02A` Z-Wave sensor and how to automatically link items to it.

Tip: Integer values can be specified either as integer (`20`) or hex (`0x14`)

The entries `thing config`, `create items` and `channels` are optional and can be combined as desired.

```
# Test mode: will not do anything but instead print out information
test: True

# Define filters which will reduce the number of things,
# all defined filters have to match for further processing
filter:
  thing_type: zwave:philio_pst02a_00_000
```

(continues on next page)

(continued from previous page)

```yaml
# Set this configuration every matching thing. HABApp will automatically only
# change the values which are not already correct.
# Here it is the z-wave parameters which are responsible for the device behaviour
thing config:
  4: 99     # Light Threshold
  5: 8      # Operation Mode
  6: 4      # MultiSensor Function Switch
  7: 20     # Customer Function

# Create items for every matching thing
create items:
 - type: Number
   name: '{thing_label, :(.+)$}_MyNumber'          # Use the label from the thing as an␣
↪input for the name,
   label: '{thing_label, :(.+)$} MyNumber [%d]'    # the regex will take everything from␣
↪the ':' on until the end
   icon: battery

channels:
  # reduce the channels of the thing with these filters
  # and link items to it
  - filter:
      channel_type: zwave:alarm_motion
    link items:
      - type: Number
        name: '{thing_label, :(.+)$}_Movement'           # Use the label from the thing␣
↪as an input for the name,
        label: '{thing_label, :(.+)$} Movement [%d %%]'  # the regex will take␣
↪everything from the ':' on until the end
        icon: battery
        groups: ['group1', 'group2']
        tags: ['tag1']

  - filter:
      channel_type: zwave:sensor_temperature
    link items:
      - type: Number
        name: '{thing_label, :(.+)$}_Temperature'
        label: '{thing_label, :(.+)$} Temperature [%d %%]'
        icon: battery
```

Multiple filters and filter definitions in one file

It is possible to add multiple thing processors into one file. To achieve this the root entry is now a list.

Filters can also be lists e.g. if the have to be applied multiple times to the same filed.

```
- test: True
  filter:
    thing_type: zwave:philio_pst02a_00_000
  ...

- test: True
  # multiple filters on the same field, all have to match
  filter:
  - thing_type: zwave:fibaro.+
  - thing_type: zwave:fibaro_fgrgbw_00_000
  ...
```

9.5.4 Thing configuration

With the `thing config` block it is possible to set a configuration for each matching thing. If the parameters are already correct, they will not be set again.

Warning: The value of the configuration parameters will not be checked and will be written as specified. It is recommended to use HABmin or PaperUI to generate the initial configuration and use this mechanism to spread it to things of the same type.

Example

```
thing config:
  4: 99     # Light Threshold
  5: 8      # Operation Mode
  6: 4      # MultiSensor Function Switch
  7: 20     # Customer Function
```

References to other parameters

It is possible to use references to mathematically build parameters from other parameters. Typically this would be fade duration and refresh interval. References to other parameter values can be created with `$`. Example:

```
thing config:
  5: 8
  6: '$5 / 2'       # Use value from parameter 5 and divide it by two.
  7: 'int($5 / 2)'  # it is possible to use normal python data conversions
```

9.5.5 Item configuration

Items can be configured under `create items -> []` and `channels -> [] -> link items -> []`.

Structure

Mandatory values are `type` and `name`, all other values are optional.

```
type: Number
name: my_name
label: my_label
icon: my_icon
groups: ['group1', 'group2']
tags: ['tag1', 'tag1']
```

Metadata

It is possible to add metadata to the created items through the optional `metadata` entry in the item config.

There are two forms how metadata can be set. The implicit form for simple key-value pairs (e.g. `autoupdate`) or the explicit form where the entries are unter `value` and `config` (e.g. `alexa`)

```
- type: Number
  name: '{thing_label, :(.+)$}_Temperature'
  label: '{thing_label, :(.+)$} Temperature [%d %%]'
  icon: battery
  metadata:
    autoupdate: 'false'
    homekit: 'TemperatureSensor'
    alexa:
      'value': 'Fan'
      'config':
        'type': 'oscillating'
        'speedSteps': 3
```

The config is equivalent to the following item configuration:

```
Number MyLabel_Temperature  "MyLabel Temperature [%d %%]" { autoupdate="false", homekit=
↪"TemperatureSensor", alexa="Fan" [ type="oscillating", speedSteps=3 ] }
```

9.5.6 Fields

Filtering things/channels

The filter value can be applied to any available field from the Thing/Channel. The filter value is a regex that has to fully match the value.

Syntax:

```
filter:
  FIELD_NAME: REGULAR_EXPRESSION
```

e.g.

```
filter:
  thing_uid: zwave:device:controller:node35
```

If multiple filters are specified all have to match to select the Thing or Channel.

```
# Multiple filters on different columns
filter:
  thing_type: zwave:fibaro.+
  thing_uid: zwave:device:controller:node35

# Multiple filters on the same columns (rarely needed)
filter:
- thing_type: zwave:fibaro.+
- thing_type: zwave:fibaro_fgrgbw_00_000
```

Field values as inputs

Filed values are available for item configuration and can be applied to all fields in the item configuration except for `type` and `metadata`.

Syntax

Macros that select field values are framed with `{}` so the containing string has to be put in annotation marks. There are three modes of operation with wildcards:

1. Just insert the value from the field:
   `{field}`
2. Insert a part of the value from the field. A regular expression is used to extract the part and therefore has to contain a capturing group.
   `{field, regex(with_group)}`
3. Do a regex replace on the value from the field and use the result
   `{field, regex, replace}`

Available fields

Tip: Test mode will show a table with all available fields and their value

The following fields are available for things:

- `thing_uid`
- `thing_type`
- `thing_location`
- `thing_label`
- `bridge_uid`

Additional available fields for channels:

- channel_uid
- channel_type
- channel_label
- channel_kind

9.5.7 Example

Log output

This will show the output for the example from *File Structure*

```
Loading /config/thing_philio.yml!
+-------------------------------------------------------------------------------------------
↪-------------------------------------------------------------------------------------------
↪---+
|                                                                                           ␣
↪Thing overview                                                                               ␣
↪        |
+---------------------------------+---------------------------+----------------+--------
↪---------------------------------+---------------------------------------------------+-------
↪---+
|           thing_uid             |         thing_type        | thing_location |       ␣
↪    thing_label                  |                 bridge_uid                    |␣
↪editable |
+---------------------------------+---------------------------+----------------+--------
↪---------------------------------+---------------------------------------------------+-------
↪---+
| zwave:device:controller:node32  | zwave:fibaro_fgrgbw_00_000 | Room1          | Fibaro␣
↪RGBW (Node 32): Room1 RGBW      | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node7   | zwave:fibaro_fgrgbw_00_000 | Room2          | Fibaro␣
↪RGBW (Node 07): Room2 RGBW      | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node23  | zwave:fibaro_fgrgbw_00_000 | Room3          | Fibaro␣
↪RGBW (Node 23): Room3 RGBW      | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node35  | zwave:philio_pst02a_00_000 | Room1          | Philio␣
↪PST02A (Node 35): Room1 Door    | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node15  | zwave:philio_pst02a_00_000 | Room2          | Philio␣
↪PST02A (Node 15): Room2 Window  | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node17  | zwave:philio_pst02a_00_000 | Room3          | Philio␣
↪PST02A (Node 17): Room3 Window  | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node3   | zwave:philio_pst02a_00_000 | Room1          | Philio␣
↪PST02A (Node 03): Room1 Window  | zwave:serial_zstick:controller               | True ␣
↪    |
| zwave:device:controller:node5   | zwave:philio_pst02a_00_000 | Room4          | Philio␣
↪PST02A (Node 05): FrontDoor     | zwave:serial_zstick:controller               | True ␣
↪    |
```

(continues on next page)

(continued from previous page)

```
| zwave:serial_zstick:controller  | zwave:serial_zstick        |                | ZWave␣
↪Controller                       |                                             |␣
↪False    |
+---------------------------------+----------------------------+----------------+--------
↪---------------------------------+---------------------------------------------+-------
↪---+
thing_type "zwave:philio_pst02a_00_000" matches for zwave:device:controller:node35!
thing_type "zwave:philio_pst02a_00_000" matches for zwave:device:controller:node15!
thing_type "zwave:philio_pst02a_00_000" matches for zwave:device:controller:node17!
thing_type "zwave:philio_pst02a_00_000" matches for zwave:device:controller:node3!
thing_type "zwave:philio_pst02a_00_000" matches for zwave:device:controller:node5!
+-------------------------------------------------------------------------------------------
↪-----------------------------------+
|                                                 Current configuration                 ␣
↪                                   |
+-------------------------+-------------------+-------------------+-------------------+--
↪----------------+------------------+
|        Parameter        | controller:node35 | controller:node15 | controller:node17 |␣
↪controller:node3 | controller:node5 |
+-------------------------+-------------------+-------------------+-------------------+--
↪----------------+------------------+
| 2                       | -1                | -1                | -1                | -
↪1               | -1               |
| 3                       | 80                | 80                | 80                |␣
↪80               | 80               |
| 4                       | 99                | 99                | 99                |␣
↪99               | 99               |
| 5                       | 0                 | 8                 | 8                 |␣
↪8                | 8                |
| 6                       | 4                 | 0                 | 0                 |␣
↪0                | 0                |
| 7                       | 22                | 20                | 20                |␣
↪20               | 20               |
| 8                       | 3                 | 3                 | 3                 |␣
↪3                | 3                |
| 9                       | 4                 | 0                 | 4                 |␣
↪4                | 4                |
| 10                      | 12                | 12                | 12                |␣
↪12               | 12               |
| 11                      | 12                | 12                | 12                |␣
↪12               | 12               |
| 12                      | 12                | 12                | 2                 |␣
↪12               | 4                |
| 13                      | 12                | 12                | 2                 |␣
↪12               | 4                |
| 20                      | 30                | 30                | 30                |␣
↪30               | 30               |
| 21                      | 1                 | 0                 | 0                 |␣
↪0                | 0                |
| 22                      | 0                 | 0                 | 0                 |␣
↪0                | 0                |
| Group1                  | ['controller']    | ['controller']    | ['controller']    | [
↪'controller']    | ['controller']    |
```

(continues on next page)

(continued from previous page)

```
| Group2                   | []                | []                | []                |␣
↪[]                | []                |
| binding_cmdrepollperiod | 1500              | 1500              | 1500              |␣
↪1500              | 1500              |
| binding_pollperiod      | 86400             | 86400             | 86400             |␣
↪86400             | 86400             |
| wakeup_interval         | 86400             | 86400             | 86400             |␣
↪86400             | 86400             |
+-------------------------+-------------------+-------------------+-------------------+--
↪----------------+------------------+
Would set {5: 8, 7: 20} for zwave:device:controller:node35
Would set {6: 4} for zwave:device:controller:node15
Would set {6: 4} for zwave:device:controller:node17
Would set {6: 4} for zwave:device:controller:node3
Would set {6: 4} for zwave:device:controller:node5
+-------------------------------------------------------------------------------------------
↪-------------------------------+
|                                        Channels for zwave:philio_pst02a_00_000         ␣
↪                               |
+-------------------------------------------------+--------------------------+---------
↪---------------+--------------+
|                    channel_uid                  |       channel_type       |    ␣
↪channel_label      | channel_kind |
+-------------------------------------------------+--------------------------+---------
↪---------------+--------------+
| zwave:device:controller:node35:sensor_door      | zwave:sensor_door        | Door/
↪Window Sensor     | STATE        |
| zwave:device:controller:node35:alarm_motion     | zwave:alarm_motion       | Motion␣
↪Sensor          | STATE        |
| zwave:device:controller:node35:alarm_tamper     | zwave:alarm_tamper       | Tamper␣
↪Alarm           | STATE        |
| zwave:device:controller:node35:sensor_luminance | zwave:sensor_luminance   | Sensor␣
↪(luminance)     | STATE        |
| zwave:device:controller:node35:sensor_temperature | zwave:sensor_temperature | Sensor␣
↪(temperature)   | STATE        |
| zwave:device:controller:node35:alarm_access     | zwave:alarm_access       | Alarm␣
↪(Access Control) | STATE        |
| zwave:device:controller:node35:alarm_burglar    | zwave:alarm_burglar      | Alarm␣
↪(Burglar)        | STATE        |
| zwave:device:controller:node35:battery-level    | system:battery-level     |␣
↪Batterieladung         | STATE        |
+-------------------------------------------------+--------------------------+---------
↪---------------+--------------+
channel_type "zwave:alarm_motion" matches for zwave:device:controller:node35:alarm_
↪motion!
channel_type "zwave:sensor_temperature" matches for␣
↪zwave:device:controller:node35:sensor_temperature!

channel_type "zwave:alarm_motion" matches for zwave:device:controller:node15:alarm_
↪motion!
channel_type "zwave:sensor_temperature" matches for␣
↪zwave:device:controller:node15:sensor_temperature!
```

(continues on next page)

(continued from previous page)

```
channel_type "zwave:alarm_motion" matches for zwave:device:controller:node17:alarm_
↪motion!
channel_type "zwave:sensor_temperature" matches for␣
↪zwave:device:controller:node17:sensor_temperature!

channel_type "zwave:alarm_motion" matches for zwave:device:controller:node3:alarm_motion!
channel_type "zwave:sensor_temperature" matches for zwave:device:controller:node3:sensor_
↪temperature!

channel_type "zwave:alarm_motion" matches for zwave:device:controller:node5:alarm_motion!
channel_type "zwave:sensor_temperature" matches for zwave:device:controller:node5:sensor_
↪temperature!

Would create Item(type='Number', name='Room1_Door_MyNumber', label='Room1 Door MyNumber [
↪%d]', icon='battery', groups=[], tags=[], link=None)
Would create Item(type='Number', name='Room1_Door_Movement', label='Room1 Door Movement [
↪%d %%]', icon='battery', groups=['group1', 'group2'], tags=['tag1'], link=
↪'zwave:device:controller:node35:alarm_motion')
Would create Item(type='Number', name='Room1_Door_Temperature', label='Room1 Door␣
↪Temperature [%d %%]', icon='battery', groups=[], tags=[], link=
↪'zwave:device:controller:node35:sensor_temperature')
Would create Item(type='Number', name='Room2_Window_MyNumber', label='Room2 Window␣
↪MyNumber [%d]', icon='battery', groups=[], tags=[], link=None)
Would create Item(type='Number', name='Room2_Window_Movement', label='Room2 Window␣
↪Movement [%d %%]', icon='battery', groups=['group1', 'group2'], tags=['tag1'], link=
↪'zwave:device:controller:node15:alarm_motion')
Would create Item(type='Number', name='Room2_Window_Temperature', label='Room2 Window␣
↪Temperature [%d %%]', icon='battery', groups=[], tags=[], link=
↪'zwave:device:controller:node15:sensor_temperature')
Would create Item(type='Number', name='Room3_Window_MyNumber', label='Room3 Window␣
↪MyNumber [%d]', icon='battery', groups=[], tags=[], link=None)
Would create Item(type='Number', name='Room3_Window_Movement', label='Room3 Window␣
↪Movement [%d %%]', icon='battery', groups=['group1', 'group2'], tags=['tag1'], link=
↪'zwave:device:controller:node17:alarm_motion')
Would create Item(type='Number', name='Room3_Window_Temperature', label='Room3 Window␣
↪Temperature [%d %%]', icon='battery', groups=[], tags=[], link=
↪'zwave:device:controller:node17:sensor_temperature')
Would create Item(type='Number', name='Room1_Window_MyNumber', label='Room1 Window␣
↪MyNumber [%d]', icon='battery', groups=[], tags=[], link=None)
Would create Item(type='Number', name='Room1_Window_Movement', label='Room1 Window␣
↪Movement [%d %%]', icon='battery', groups=['group1', 'group2'], tags=['tag1'], link=
↪'zwave:device:controller:node3:alarm_motion')
Would create Item(type='Number', name='Room1_Window_Temperature', label='Room1 Window␣
↪Temperature [%d %%]', icon='battery', groups=[], tags=[], link=
↪'zwave:device:controller:node3:sensor_temperature')
Would create Item(type='Number', name='FrontDoor_MyNumber', label='FrontDoor MyNumber [
↪%d]', icon='battery', groups=[], tags=[], link=None)
Would create Item(type='Number', name='FrontDoor_Movement', label='FrontDoor Movement [
↪%d %%]', icon='battery', groups=['group1', 'group2'], tags=['tag1'], link=
↪'zwave:device:controller:node5:alarm_motion')
Would create Item(type='Number', name='FrontDoor_Temperature', label='FrontDoor␣
↪Temperature [%d %%]', icon='battery', groups=[], tags=[], link=
↪'zwave:device:controller:node5:sensor_temperature')
```

(continues on next page)

(continued from previous page)

Created items file

```
Number   Room1_Door_MyNumber          "Room1 Door MyNumber [%d]"              <battery>
Number   Room1_Door_Movement          "Room1 Door Movement [%d %%]"           <battery>   ␣
↪(group1, group2)     [tag1]   {channel = "zwave:device:controller:node35:alarm_motion"}
Number   Room1_Door_Temperature       "Room1 Door Temperature [%d %%]"        <battery>   ␣
↪                                {channel = "zwave:device:controller:node35:sensor_
↪temperature"}
Number   Room2_Window_MyNumber        "Room2 Window MyNumber [%d]"            <battery>
Number   Room2_Window_Movement        "Room2 Window Movement [%d %%]"         <battery>   ␣
↪(group1, group2)     [tag1]   {channel = "zwave:device:controller:node15:alarm_motion"}
Number   Room2_Window_Temperature     "Room2 Window Temperature [%d %%]"      <battery>   ␣
↪                                {channel = "zwave:device:controller:node15:sensor_
↪temperature"}
Number   Room3_Window_MyNumber        "Room3 Window MyNumber [%d]"            <battery>
Number   Room3_Window_Movement        "Room3 Window Movement [%d %%]"         <battery>   ␣
↪(group1, group2)     [tag1]   {channel = "zwave:device:controller:node17:alarm_motion"}
Number   Room3_Window_Temperature     "Room3 Window Temperature [%d %%]"      <battery>   ␣
↪                                {channel = "zwave:device:controller:node17:sensor_
↪temperature"}
Number   Room1_Window_MyNumber        "Room1 Window MyNumber [%d]"            <battery>
Number   Room1_Window_Movement        "Room1 Window Movement [%d %%]"         <battery>   ␣
↪(group1, group2)     [tag1]   {channel = "zwave:device:controller:node3:alarm_motion"}
Number   Room1_Window_Temperature     "Room1 Window Temperature [%d %%]"      <battery>   ␣
↪                                {channel = "zwave:device:controller:node3:sensor_
↪temperature"}
Number   FrontDoor_MyNumber           "FrontDoor MyNumber [%d]"               <battery>
Number   FrontDoor_Movement           "FrontDoor Movement [%d %%]"            <battery>   ␣
↪(group1, group2)     [tag1]   {channel = "zwave:device:controller:node5:alarm_motion"}
Number   FrontDoor_Temperature        "FrontDoor Temperature [%d %%]"         <battery>   ␣
↪                                {channel = "zwave:device:controller:node5:sensor_
↪temperature"}
```

9.6 Example openHAB rules

9.6.1 Example 1

```
import HABApp
from HABApp.core.events import ValueUpdateEvent, ValueChangeEvent
from HABApp.openhab.events import ItemStateEvent, ItemCommandEvent, ItemStateChangedEvent
from HABApp.openhab.items import SwitchItem, ContactItem, DatetimeItem


class MyOpenhabRule(HABApp.Rule):

    def __init__(self):
```

(continues on next page)

(continued from previous page)

```
        super().__init__()

        # get items
        test_contact = ContactItem.get_item('TestContact')
        test_date_time = DatetimeItem.get_item('TestDateTime')
        test_switch = SwitchItem.get_item('TestSwitch')

        # Trigger on item updates
        test_contact.listen_event(self.item_state_update, ItemStateEvent)
        test_date_time.listen_event(self.item_state_update, ValueUpdateEvent)

        # Trigger on item changes
        test_contact.listen_event(self.item_state_change, ItemStateChangedEvent)
        test_date_time.listen_event(self.item_state_change, ValueChangeEvent)

        # Trigger on item commands
        test_switch.listen_event(self.item_command, ItemCommandEvent)

    def item_state_update(self, event):
        assert isinstance(event, ValueUpdateEvent)
        print( f'{event}')

    def item_state_change(self, event):
        assert isinstance(event, ValueChangeEvent)
        print( f'{event}')

        # interaction is available through self.openhab or self.oh
        self.openhab.send_command('TestItemCommand', 'ON')

        # example for interaction with openhab item type
        switch_item = SwitchItem.get_item('TestSwitch')
        if switch_item.is_on():
            switch_item.off()

    def item_command(self, event):
        assert isinstance(event, ItemCommandEvent)
        print( f'{event}')

        # interaction is available through self.openhab or self.oh
        self.oh.post_update('ReceivedCommand', str(event))


MyOpenhabRule()
```

9.6.2 Check status of things

This rule prints the status of all Things and shows how to subscribe to events of the Thing status

```
import HABApp
from HABApp import Rule
from HABApp.openhab.events import ThingStatusInfoChangedEvent
from HABApp.openhab.items import Thing


class CheckAllThings(Rule):
    def __init__(self):
        super().__init__()

        for thing in HABApp.core.Items.get_all_items():
            if isinstance(thing, Thing):
                thing.listen_event(self.thing_status_changed,␣
↪ThingStatusInfoChangedEvent)
                print(f'{thing.name}: {thing.status}')

    def thing_status_changed(self, event: ThingStatusInfoChangedEvent):
        print(f'{event.name} changed from {event.old_status} to {event.status}')


CheckAllThings()
```

9.6.3 Check status if thing is constant

Sometimes Things recover automatically from small outages. This rule only triggers then the Thing is constant for 60 seconds.

```
from HABApp import Rule
from HABApp.core.events import ItemNoChangeEvent
from HABApp.openhab.items import Thing


class CheckThing(Rule):
    def __init__(self, name: str):
        super().__init__()

        self.thing = Thing.get_item(name)
        watcher = self.thing.watch_change(60)
        self.thing.listen_event(self.thing_no_change, watcher.EVENT)

    def thing_no_change(self, event: ItemNoChangeEvent):
        print(f'Thing {event.name} constant for {event.seconds}')
        print(f'Status: {self.thing.status}')


CheckThing('my:thing:uid')
```

```
Thing test_watch constant for 60
Status: ONLINE
```

CHAPTER

TEN

MQTT

10.1 Interaction with the MQTT broker

Interaction with the MQTT broker is done through the `self.mqtt` object in the rule or through the *MqttItem*. When receiving a topic for the first time a new *MqttItem* will automatically be created.

10.2 Rule Interface

class mqtt

publish(*topic: str*, *payload: typing.Any*[, *qos: int = None*, *retain: bool = None*]) → int
Publish a value under a certain topic.

Parameters

- **topic** – MQTT topic
- **payload** – MQTT Payload
- **qos** (*int*) – QoS, can be 0, 1 or 2. If not specified value from configuration file will be used.
- **retain** (*bool*) – retain message. If not specified value from configuration file will be used.

Returns 0 if successful

subscribe(*self*, *topic: str*[, *qos: int = None*]) → int
Subscribe to a MQTT topic. Subscriptions will be active until next disconnect. For persistent subscriptions use the configuration file

Parameters

- **topic** – MQTT topic to subscribe to
- **qos** – QoS, can be 0, 1 or 2. If not specified value from configuration file will be used.

Returns 0 if successful

unsubscribe(*self*, *topic: str*) → int
Unsubscribe from a MQTT topic

Parameters **topic** – MQTT topic

Returns 0 if successful

10.3 Mqtt item types

Mqtt items have an additional publish method which make interaction with the mqtt broker easier.

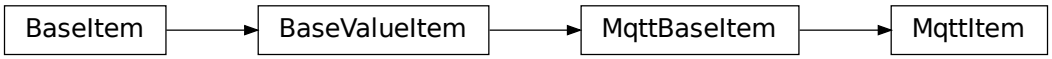
```
from HABApp.mqtt.items import MqttItem

# items can be created manually or will be automatically
# created when the first mqtt message is received
my_mqtt_item = MqttItem.get_create_item('test/topic')

# easy to publish values
my_mqtt_item.publish('new_value')

# comparing the item to get the state works, too
if my_mqtt_item == 'test':
    pass # do something
```

10.3.1 MqttItem



class HABApp.mqtt.items.**MqttItem**(*name*, *initial_value=None*)

A simple item that represents a topic and a value

classmethod get_create_item(*name*, *initial_value=None*)

Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

- **name** (str) – item name
- **initial_value** – state the item will have if it gets created

Return type *MqttItem*

Returns item

classmethod get_item(*name*)

Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_value(*default_value=None*)

Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

publish(*payload*, *qos=None*, *retain=None*)
Publish the payload under the topic from the item.

Parameters

- **payload** – MQTT Payload
- **qos** (Optional[int]) – QoS, can be 0, 1 or 2. If not specified value from configuration file will be used.
- **retain** (Optional[bool]) – retain message. If not specified value from configuration file will be used.

Returns 0 if successful

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

Return type str

Returns Name of the item (read only)

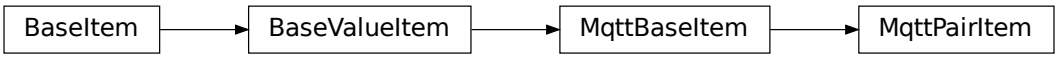
10.3.2 MqttPairItem

An item that consolidates a topic that reports states from a device and a topic that is used to write to a device. It is created on the topic that reports the state from the device.

```
from HABApp.mqtt.items import MqttPairItem

# MqttPairItem works out of the box with zigbee2mqtt
mqtt = MqttPairItem.get_create_item("zigbee2mqtt/my_bulb/brightness")
mqtt.publish("255")  # <-- will use the write topic

# equivalent to
mqtt = MqttPairItem.get_create_item("zigbee2mqtt/my_bulb/brightness", write_topic=
↪"zigbee2mqtt/my_bulb/set/brightness")
```



class HABApp.mqtt.items.**MqttPairItem**(*name*, *initial_value=None*, *write_topic=None*)

An item that represents both a topic that is used to read and a corresponding topic that is used to write values

classmethod get_create_item(*name*, *write_topic=None*, *initial_value=None*)

Creates a new item in HABApp and returns it or returns the already existing one with the given name. HABApp tries to automatically derive the write topic from the item name. In cases where this does not work it can be specified manually.

Parameters

- **name** (str) – item name (topic that reports the state)

- **write_topic** (Optional[str]) – topic that is used to write values or None (default) to build it automatically
- **initial_value** – state the item will have if it gets created

Return type *MqttPairItem*

Returns item

classmethod get_item(*name*)
Returns an already existing item. If it does not exist or has a different item type an exception will occur.

Parameters **name** (str) – Name of the item

get_value(*default_value=None*)
Return the value of the item.

Parameters **default_value** – Return this value if the item value is None

Return type Any

Returns value of the item

listen_event(*callback*, *event_type*)
Register an event listener which listens to all event that the item receives

Parameters

- **callback** (Callable[[Any], Any]) – callback that accepts one parameter which will contain the event
- **event_type** (Union[AllEvents, *EventFilter*, Any]) – Event filter. This is typically ValueUpdateEvent or ValueChangeEvent which will also trigger on changes/update from openHAB or mqtt.

Return type EventBusListener

post_value(*new_value*)
Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

publish(*payload*, *qos=None*, *retain=None*)
Publish the payload under the write topic from the item.

Parameters

- **payload** – MQTT Payload
- **qos** (Optional[int]) – QoS, can be 0, 1 or 2. If not specified value from configuration file will be used.
- **retain** (Optional[bool]) – retain message. If not specified value from configuration file will be used.

Returns 0 if successful

set_value(*new_value*)
Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

watch_change(*secs*)
Generate an event if the item does not change for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoChangeWatch*

Returns The watch obj which can be used to cancel the watch

watch_update(*secs*)
Generate an event if the item does not receive and update for a certain period of time. Has to be called from inside a rule function.

Parameters **secs** (Union[int, float, timedelta]) – secs after which the event will occur, max 1 decimal digit for floats

Return type *ItemNoUpdateWatch*

Returns The watch obj which can be used to cancel the watch

property last_change: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been changed (read only)

property last_update: pendulum.datetime.DateTime

Return type DateTime

Returns Timestamp of the last time when the item has been updated (read only)

property name: str

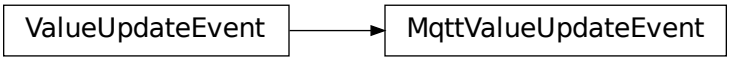
Return type str

Returns Name of the item (read only)

10.4 Mqtt event types

10.4.1 MqttValueUpdateEvent

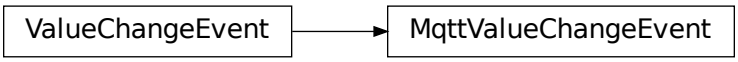
Since this event inherits from *ValueUpdateEvent* you can listen to *ValueUpdateEvent* and it will also trigger for *MqttValueUpdateEvent*.



class HABApp.mqtt.events.**MqttValueUpdateEvent**(*name=None*, *value=None*)

10.4.2 MqttValueChangeEvent

Since this event inherits from *ValueChangeEvent* you can listen to *ValueChangeEvent* and it will also trigger for *MqttValueUpdateEvent*.



class HABApp.mqtt.events.**MqttValueChangeEvent**(*name=None*, *value=None*, *old_value=None*)

10.5 Example MQTT rule

```
import datetime
import random

import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.mqtt.items import MqttItem


class ExampleMqttTestRule(HABApp.Rule):
    def __init__(self):
        super().__init__()

        self.run.every(
            start_time=datetime.timedelta(seconds=10),
            interval=datetime.timedelta(seconds=20),
            callback=self.publish_rand_value
        )

        self.my_mqtt_item = MqttItem.get_create_item('test/test')

        self.listen_event('test/test', self.topic_updated, ValueUpdateEvent)

    def publish_rand_value(self):
        print('test mqtt_publish')
        self.my_mqtt_item.publish(str(random.randint(0, 1000)))

    def topic_updated(self, event):
        assert isinstance(event, ValueUpdateEvent), type(event)
        print( f'mqtt topic "test/test" updated to {event.value}')


ExampleMqttTestRule()
```

CHAPTER

ELEVEN

ADVANCED USAGE

11.1 HABApp Topics

There are several internal topics which can be used to react to HABApp changes from within rules. An example would be dynamically reloading files or an own notifier in case there are errors (e.g. Pushover).

Topic	Description	Events
HABApp.Files	The corresponding events trigger a load/unload of the file specified in the event	*RequestFileLoadEvent* and *RequestFileUnloadEvent*
HABApp.Infos	All infos in functions and rules of HABApp create an according event	`str`
HABApp.Warnings	All warnings in functions (e.g. caught exceptions) and rules of HABApp create an according event	*HABAppException* or `str`
HABApp.Errors	All errors in functions and rules of HABApp create an according event. Use this topic to create an own notifier in case of errors (e.g. Pushover).	*HABAppException* or `str`

class HABApp.core.events.habapp_events.**RequestFileLoadEvent**(*name*)
Request (re-) loading of the specified file

Variables **filename** (*str*) – relative filename

class HABApp.core.events.habapp_events.**RequestFileUnloadEvent**(*name*)
Request unloading of the specified file

Variables **filename** (*str*) – relative filename

class HABApp.core.events.habapp_events.**HABAppException**(*func_name*, *exception*, *traceback*)
Contains information about an Exception that has occurred in HABApp

Variables

- **func_name** (*str*) – name of the function where the error occurred
- **traceback** (*str*) – traceback
- **exception** (*Exception*) – Exception

to_str()
Create a readable str with all information

Return type `str`

11.2 File properties

For every HABApp file it is possible to specify some properties. The properties are specified as a comment (prefixed with #) somewhere at the beginning of the file and are in the yml format. They keyword `HABApp` can be arbitrarily intended.

Hint: File names are not absolute but relative with a folder specific prefix. It's best to use the file name from the *RequestFileLoadEvent* from the HABApp event bus.

Configuration format

```
HABApp:
  depends on:
   - filename
  reloads on:
   - filename
```

Property	Description
depends on	The file will only get loaded when **all** of the files specified as dependencies have been successfully loaded
reloads on	The file will get automatically reloaded when **one of** the files specified will be reloaded

Example

```
# Some other stuff
#
# HABApp:
#   depends on:
#    - rules/rule_file.py
#   reloads on:
#    - params/param_file.yml

import HABApp
...
```

11.3 Invoking OpenHAB actions

The openhab REST interface does not expose actions, and thus there is no way to trigger them from HABApp. If it is not possible to create and OpenHAB item that directly triggers the action there is a way to work around it with additional items within openhab. An additional OpenHAB (note not HABapp) rule listens to changes on those items and invokes the appropriate openhab actions. On the HABApp side these actions are indirectly executed by setting the values for those items.

Below is an example how to invoke the openhab Audio and Voice actions.

First, define couple items to accept values from HABApp, and place them in /etc/openhab2/items/habapp-bridge.items:

```
String AudioVoiceSinkName
```

(continues on next page)

(continued from previous page)

```
String TextToSpeechMessage
String AudioFileLocation
String AudioStreamUrl
```

Second, create the JSR223 script to invoke the actions upon changes in the values of the items above.

```
from core import osgi
from core.jsr223 import scope
from core.rules import rule
from core.triggers import when
from org.eclipse.smarthome.model.script.actions import Audio
from org.eclipse.smarthome.model.script.actions import Voice

SINK_ITEM_NAME = 'AudioVoiceSinkName'

@rule("Play voice TTS message")
@when("Item TextToSpeechMessage changed")
def onTextToSpeechMessageChanged(event):
    ttl = scope.items[event.itemName].toString()
    if ttl is not None and ttl != '':
        Voice.say(ttl, None, scope.items[SINK_ITEM_NAME].toString())

        # reset the item to wait for the next message.
        scope.events.sendCommand(event.itemName, '')

@rule("Play audio stream URL")
@when("Item AudioStreamUrl changed")
def onTextToSpeechMessageChanged(event):
    stream_url = scope.items[event.itemName].toString()
    if stream_url is not None and stream_url != '':
        Audio.playStream(scope.items[SINK_ITEM_NAME].toString(), stream_url)

        # reset the item to wait for the next message.
        scope.events.sendCommand(event.itemName, '')

@rule("Play local audio file")
@when("Item AudioFileLocation changed")
def onTextToSpeechMessageChanged(event):
    file_location = scope.items[event.itemName].toString()
    if file_location is not None and file_location != '':
        Audio.playSound(scope.items[SINK_ITEM_NAME].toString(), file_location)

        # reset the item to wait for the next message.
        scope.events.sendCommand(event.itemName, '')
```

Finally, define the HABApp functions to indirectly invoke the actions:

```
def play_local_audio_file(sink_name: str, file_location: str):
    """ Plays a local audio file on the given audio sink. """
    HABApp.openhab.interface.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface.send_command(ACTION_AUDIO_LOCAL_FILE_LOCATION_ITEM_NAME,␣
↪file_location)
```

(continues on next page)

(continued from previous page)

```
def play_stream_url(sink_name: str, url: str):
    """ Plays a stream URL on the given audio sink. """
    HABApp.openhab.interface.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface.send_command(ACTION_AUDIO_STREAM_URL_ITEM_NAME, url)


def play_text_to_speech_message(sink_name: str, tts: str):
    """ Plays a text to speech message on the given audio sink. """
    HABApp.openhab.interface.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface.send_command(ACTION_TEXT_TO_SPEECH_MESSAGE_ITEM_NAME, tts)
```

11.4 Mocking OpenHAB items and events for tests

It is possible to create mock items in HABApp which do not exist in Openhab to create unit tests for rules and libraries. Ensure that this mechanism is only used for testing because since the items will not exist in openhab they will not get updated which can lead to hard to track down errors.

Examples:

Add an openhab mock item to the item registry

```
import HABApp
from HABApp.openhab.items import SwitchItem

item = SwitchItem('my_switch', 'ON')
HABApp.core.Items.add_item(item)
```

Remove the mock item from the registry

```
HABApp.core.Items.pop_item('my_switch')
```

Note that there are some item methods that encapsulate communication with openhab (e.g.: `SwitchItem.on()`, `SwithItem.off()`, and `DimmerItem.percentage()`) These currently do not work with the mock items. The state has to be changed like any internal item.

```
import HABApp
from HABApp.openhab.items import SwitchItem
from HABApp.openhab.definitions import OnOffValue

item = SwitchItem('my_switch', 'ON')
HABApp.core.Items.add_item(item)

item.set_value(OnOffValue.ON)    # without bus event
item.post_value(OnOffValue.OFF)  # with bus event
```

CHAPTER

TWELVE

ASYNCIO

Warning:

Please make sure you know what you are doing when using async functions!
If you have no asyncio experience please do not use this! The use of blocking calls in async functions will prevent HABApp from working properly!

12.1 async http

Async http calls are available through the `self.async_http` object in rule instances.

12.1.1 Functions

class HABApp.rule.interfaces.**AsyncHttpConnection**

get(*url*, *params=None*, ***kwargs*)
http get request

Parameters

- **url** (str) – Request URL
- **params** (Optional[Mapping[str, str]]) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **data** – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **json** – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **kwargs** (Any) – See aiohttp request for further possible kwargs

Return type _RequestContextManager

Returns awaitable

post(*url*, *params=None*, *data=None*, *json=None*, ***kwargs*)
http post request

Parameters

- **url** (str) – Request URL

- **params** (Optional[Mapping[str, str]]) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **data** (Optional[Any]) – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **json** (Optional[Any]) – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **kwargs** (Any) – See aiohttp request for further possible kwargs

Return type _RequestContextManager

Returns awaitable

put(*url*, *params=None*, *data=None*, *json=None*, ***kwargs*)
http put request

Parameters

- **url** (str) – Request URL
- **params** (Optional[Mapping[str, str]]) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **data** (Optional[Any]) – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **json** (Optional[Any]) – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **kwargs** (Any) – See aiohttp request for further possible kwargs

Return type _RequestContextManager

Returns awaitable

get_client_session()
Return the aiohttp client session object for use in aiohttp libraries

Return type ClientSession

Returns session object

12.1.2 Examples

```
import asyncio

import HABApp


class AsyncRule(HABApp.Rule):

    def __init__(self):
        super().__init__()

        self.run.soon(self.async_func)

    async def async_func(self):
        await asyncio.sleep(2)
```

(continues on next page)

(continued from previous page)

```
        async with self.async_http.get('http://httpbin.org/get') as resp:
            print(resp)
            print(await resp.text())


AsyncRule()
```

CHAPTER

THIRTEEN

UTIL - HELPERS AND UTILITIES

The util package contains useful classes which make rule creation easier.

13.1 Functions

13.1.1 min

This function is very useful together with the all possible functions of *ValueMode* for the *MultiModeItem*. For example it can be used to automatically disable or calculate the new value of the *ValueMode* It behaves like the standard python function except that it will ignore `None` values which are sometimes set as the item state.

```
from HABApp.util.functions import min

print(min(1, 2, None))
```

HABApp.util.functions.**min**(**args*, *default=None*)

Behaves like the built in min function but ignores any `None` values. e.g. `min([1, None, 2]) == 1`. If the iterable is empty `default` will be returned.

Parameters

- **args** – Single iterable or 1..n arguments
- **default** – Value that will be returned if the iterable is empty

Returns min value

13.1.2 max

This function is very useful together with the all possible functions of *ValueMode* for the *MultiModeItem*. For example it can be used to automatically disable or calculate the new value of the *ValueMode* It behaves like the standard python function except that it will ignore `None` values which are sometimes set as the item state.

```
from HABApp.util.functions import max

print(max(1, 2, None))
```

HABApp.util.functions.**max**(**args*, *default=None*)

Behaves like the built in max function but ignores any `None` values. e.g. `max([1, None, 2]) == 2`. If the iterable is empty `default` will be returned.

Parameters

- **args** – Single iterable or 1..n arguments
- **default** – Value that will be returned if the iterable is empty

Returns max value

13.1.3 rgb_to_hsb

Converts a rgb value to hsb color space

```
from HABApp.util.functions import rgb_to_hsb

print(rgb_to_hsb(224, 201, 219))
```

HABApp.util.functions.**rgb_to_hsb**(*r*, *g*, *b*, *max_rgb_value=255*, *ndigits=2*)

Convert from rgb to hsb/hsv

Parameters

- **r** (Union[int, float]) – red value
- **g** (Union[int, float]) – green value
- **b** (Union[int, float]) – blue value
- **max_rgb_value** (int) – maximal possible rgb value (e.g. 255 for 8 bit or 65.535 for 16bit values)
- **ndigits** (Optional[int]) – Round the hsb values to the specified digits, None to disable rounding

Return type Tuple[float, float, float]

Returns Values for hue, saturation and brightness / value

13.1.4 hsb_to_rgb

Converts a hsb value to the rgb color space

```
from HABApp.util.functions import hsb_to_rgb

print(hsb_to_rgb(150, 40, 100))
```

HABApp.util.functions.**hsb_to_rgb**(*h*, *s*, *b*, *max_rgb_value=255*)

Convert from rgb to hsv/hsb

Parameters

- **h** – hue
- **s** – saturation
- **b** – brightness / value
- **max_rgb_value** – maximal value for the returned rgb values (e.g. 255 for 8 bit or 65.535 16bit values)

Return type Tuple[int, int, int]

Returns Values for red, green and blue

13.2 CounterItem

13.2.1 Example

```
from HABApp.util import CounterItem
c = CounterItem.get_create_item('MyCounter', initial_value=5)
print(c.increase())
print(c.decrease())
print(c.reset())
```

```
6
5
5
```

13.2.2 Documentation

class HABApp.util.**CounterItem**(*name*, *initial_value=0*)

Implements a simple thread safe counter

Parameters **initial_value** (int) – Initial value of the counter

__init__(*name*, *initial_value=0*)

Parameters **initial_value** (int) – Initial value of the counter

set_value(*new_value*)

Set a new value without creating events on the event bus

Parameters **new_value** – new value of the item

Return type bool

Returns True if state has changed

post_value(*new_value*)

Set a new value and post appropriate events on the HABApp event bus (ValueUpdateEvent, ValueChangeEvent)

Parameters **new_value** – new value of the item

Returns True if state has changed

reset()

Reset value to initial value

increase(*step=1*)

Increase value

Parameters **step** – increase by this value, default = 1

Return type int

Returns value of the counter

decrease(*step=1*)

Decrease value

Parameters **step** – decrease by this value, default = 1

Return type int

Returns value of the counter

13.3 Statistics

13.3.1 Example

```
s = Statistics(max_samples=4)
for i in range(1,4):
    s.add_value(i)
    print(s)
```

```
<Statistics sum: 1.0, min: 1.00, max: 1.00, mean: 1.00, median: 1.00>
<Statistics sum: 3.0, min: 1.00, max: 2.00, mean: 1.50, median: 1.50>
<Statistics sum: 6.0, min: 1.00, max: 3.00, mean: 2.00, median: 2.00>
```

13.3.2 Documentation

class HABApp.util.**Statistics**(*max_age=None*, *max_samples=None*)

Calculate mathematical statistics of numerical values.

Variables

- **sum** – sum of all values
- **min** – minimum of all values
- **max** – maximum of all values
- **mean** – mean of all values
- **median** – median of all values
- **last_value** – last added value
- **last_change** – timestamp the last time a value was added

Parameters

- **max_age** – Maximum age of values in seconds
- **max_samples** – Maximum amount of samples which will be kept

update()

update values without adding a new value

add_value(*value*)

Add a new value and recalculate statistical values

Parameters **value** – new value

13.4 MultiModeItem

Prioritizer item which automatically switches between values with different priorities. Very useful when different states or modes overlap, e.g. automatic and manual mode. etc.

13.4.1 Basic Example

```
import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.util.multimode import MultiModeItem, ValueMode

class MyMultiModeItemTestRule(HABApp.Rule):
    def __init__(self):
        super().__init__()

        # create a new MultiModeItem
        item = MultiModeItem.get_create_item('MultiModeTestItem')
        item.listen_event(self.item_update, ValueUpdateEvent)

        # create two different modes which we will use and add them to the item
        auto = ValueMode('Automatic', initial_value=5)
        manu = ValueMode('Manual', initial_value=0)
        item.add_mode(0, auto).add_mode(10, manu)

        # This shows how to enable/disable a mode and how to get a mode from the item
        print('disable/enable the higher priority mode')
        item.get_mode('manual').set_enabled(False)  # disable mode
        item.get_mode('manual').set_value(11)       # setting a value will enable it␣
↪again

        # This shows that changes of the lower priority is only show when
        # the mode with the higher priority gets disabled
        print('')
        print('Set value of lower priority')
        auto.set_value(55)
        print('Disable higher priority')
        manu.set_enabled(False)

    def item_update(self, event):
        print(f'State: {event.value}')

MyMultiModeItemTestRule()
```

```
disable/enable the higher priority mode
State: 5
State: 11

Set value of lower priority
State: 11
Disable higher priority
State: 55
```

13.4.2 Advanced Example

```
import logging
import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.util.multimode import MultiModeItem, ValueMode

class MyMultiModeItemTestRule(HABApp.Rule):
    def __init__(self):
        super().__init__()

        # create a new MultiModeItem
        item = MultiModeItem.get_create_item('MultiModeTestItem')
        item.listen_event(self.item_update, ValueUpdateEvent)

        # helper to print the heading so we have a nice outpt
        def print_heading(_heading):
            print('')
            print('-' * 80)
            print(_heading)
            print('-' * 80)
            for p, m in item.all_modes():
                print(f'Prio {p:2d}: {m}')
            print('')


        log = logging.getLogger('AdvancedMultiMode')

        # create modes and add them
        auto = ValueMode('Automatic', initial_value=5, logger=log)
        manu = ValueMode('Manual', initial_value=10, logger=log)
        item.add_mode(0, auto).add_mode(10, manu)


        # it is possible to automatically disable a mode
        # this will disable the manual mode if the automatic mode
        # sets a value greater equal manual mode
        print_heading('Automatically disable mode')

        # A custom function can also disable the mode:
        manu.auto_disable_func = lambda low, own: low >= own

        auto.set_value(11) # <-- manual now gets disabled because
        auto.set_value(4)  #     the lower priority value is >= itself


        # It is possible to use functions to calculate the new value for a mode.
        # E.g. shutter control and the manual mode moves the shades. If it's dark the␣
↪automatic
        # mode closes the shutter again. This could be achievied by automatically␣
↪disable the
        # manual mode or if the state should be remembered then the max function should␣
↪be used
```

(continues on next page)

(continued from previous page)

```
        # create a move and use the max function for output calculation
        manu = ValueMode('Manual', initial_value=5, logger=log, calc_value_func=max)
        item.add_mode(10, manu)    # overwrite the earlier added mode

        print_heading('Use of functions')

        auto.set_value(7)   # manu uses max, so the value from auto is used
        auto.set_value(3)

    def item_update(self, event):
        print(f'Item value: {event.value}')

MyMultiModeItemTestRule()
```

```
--------------------------------------------------------------------------------
Automatically disable mode
--------------------------------------------------------------------------------
Prio  0: <ValueMode Automatic enabled: True, value: 5>
Prio 10: <ValueMode Manual enabled: True, value: 10>

[AdvancedMultiMode]     INFO | [x] Automatic: 11
[AdvancedMultiMode]     INFO | [ ] Manual (function)
Item value: 11
[AdvancedMultiMode]     INFO | [x] Automatic: 4
Item value: 4

--------------------------------------------------------------------------------
Use of functions
--------------------------------------------------------------------------------
Prio  0: <ValueMode Automatic enabled: True, value: 4>
Prio 10: <ValueMode Manual enabled: True, value: 5>

[AdvancedMultiMode]     INFO | [x] Automatic: 7
Item value: 7
[AdvancedMultiMode]     INFO | [x] Automatic: 3
Item value: 5
```

13.4.3 Example SwitchItemValueMode

The SwitchItemMode is same as ValueMode but enabled/disabled of the mode is controlled by a OpenHAB *SwitchItem*. This is very useful if the mode shall be deactivated from the OpenHAB sitemaps.

```
import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.openhab.items import SwitchItem
from HABApp.util.multimode import MultiModeItem, SwitchItemValueMode, ValueMode

class MyMultiModeItemTestRule(HABApp.Rule):
    def __init__(self):
```

(continues on next page)

(continued from previous page)

```
        super().__init__()

        # create a new MultiModeItem
        item = MultiModeItem.get_create_item('MultiModeTestItem')

        # this switch allows to enable/disable the mode
        switch = SwitchItem.get_item('Automatic_Enabled')
        print(f'Switch is {switch}')

        # this is how the switch gets linked to the mode
        # if the switch is on, the mode is on, too
        mode = SwitchItemValueMode('Automatic', switch)
        print(mode)

        # Use invert_switch if the desired behaviour is
        # if the switch is off, the mode is on
        mode = SwitchItemValueMode('AutomaticOff', switch, invert_switch=True)
        print(mode)

        # This shows how the SwitchItemValueMode can be used to disable any logic except␣
↪for the manual mode.
        # Now everything can be enabled/disabled from the openhab sitemap
        item.add_mode(100, mode)
        item.add_mode(101, ValueMode('Manual'))

MyMultiModeItemTestRule()
```

```
Switch is ON
<SwitchItemValueMode Automatic enabled: True, value: None>
<SwitchItemValueMode AutomaticOff enabled: False, value: None>
```

13.4.4 Documentation

MultiModeItem

class HABApp.util.multimode.**MultiModeItem**(*name*, *initial_value=None*)

Prioritizer *Item*

Variables **logger** – Assign a logger to get log messages about the different modes

classmethod **get_create_item**(*name*, *logger=None*, *initial_value=None*)

Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

- **name** (str) – item name
- **initial_value** – state the item will have if it gets created

Returns item

remove_mode(*name*)

Remove mode if it exists

Parameters **name** (str) – name of the mode (case insensitive)

Return type bool

Returns True if something was removed, False if nothign was found

add_mode(*priority*, *mode*)
Add a new mode to the item, if it already exists it will be overwritten

Parameters

- **priority** (int) – priority of the mode
- **mode** (BaseMode) – instance of the MultiMode class

Return type *MultiModeItem*

all_modes()
Returns a sorted list containing tuples with the priority and the mode

Return type List[Tuple[int, BaseMode]]

Returns List with priorities and modes

get_mode(*name*)
Returns a created mode

Parameters **name** (str) – name of the mode (case insensitive)

Return type BaseMode

Returns The requested MultiModeValue

calculate_value()
Recalculate the output value and post the state to the event bus (if it is not None)

Return type Any

Returns new value

ValueMode

class HABApp.util.multimode.**ValueMode**(*name*, *initial_value=None*, *enabled=None*, *enable_on_value=True*, *logger=None*, *auto_disable_after=None*, *auto_disable_func=None*, *calc_value_func=None*)

Variables

- **last_update** (*datetime.datetime*) – Timestamp of the last update/enable of this value
- **auto_disable_after** (*typing.Optional[datetime.timedelta]*) – Automatically disable this mode after a given timedelta on the next recalculation
- **auto_disable_func** (*typing.Optional[typing.Callable[[typing.Any, typing.Any], bool]]*) – Function which can be used to disable this mode. Any function that accepts two Arguments can be used. First arg is value with lower priority, second argument is own value. Return True to disable this mode.
- **calc_value_func** (*typing.Optional[typing.Callable[[typing.Any, typing.Any], typing.Any]]*) – Function to calculate the new value (e.g. min or max). Any function that accepts two Arguments can be used. First arg is value with lower priority, second argument is own value.

Parameters

- **name** (str) – Name of the mode

- **initial_value** – initial value of the mode
- **enabled** (Optional[bool]) – initial enabled state of the mode
- **enable_on_value** (bool) – If True (default) setting a value (that is not None) will also enable the mode
- **logger** (Optional[Logger]) – logger to log events
- **auto_disable_after** – see variables
- **auto_disable_func** – see variables
- **calc_value_func** – see variables

property value
Returns the current value

property enabled: bool
Returns if the value is enabled

Return type bool

set_value(*value*, *only_on_change=False*)
Set new value and recalculate overall value. If enable_on_value is set, setting a value will also enable the mode.

Parameters

- **value** – new value
- **only_on_change** (bool) – will set/enable the mode only if value differs or the mode is disabled

set_enabled(*value*, *only_on_change=False*)
Enable or disable this value and recalculate overall value

Parameters

- **value** (bool) – True/False
- **only_on_change** (bool) – enable only on change

Returns

cancel()
Remove the mode from the parent MultiModeItem and stop processing it

SwitchItemValueMode

class HABApp.util.multimode.**SwitchItemValueMode**(*name*, *switch_item*, *invert_switch=False*, *initial_value=None*, *logger=None*, *auto_disable_after=None*, *auto_disable_func=None*, *calc_value_func=None*)
SwitchItemMode, same as ValueMode but enabled/disabled of the mode is controlled by a OpenHAB *SwitchItem*

Variables

- **last_update** (*datetime.datetime*) – Timestamp of the last update/enable of this value
- **auto_disable_after** (*typing.Optional[datetime.timedelta]*) – Automatically disable this mode after a given timedelta on the next recalculation

- **auto_disable_func** (*typing.Optional[typing.Callable[[typing.Any, typing.Any], bool]]*) – Function which can be used to disable this mode. Any function that accepts two Arguments can be used. First arg is value with lower priority, second argument is own value. Return `True` to disable this mode.
- **calc_value_func** (*typing.Optional[typing.Callable[[typing.Any, typing.Any], typing.Any]]*) – Function to calculate the new value (e.g. `min` or `max`). Any function that accepts two Arguments can be used. First arg is value with lower priority, second argument is own value.

Parameters

- **name** (`str`) – Name of the mode
- **switch_item** (*SwitchItem*) – *SwitchItem* that will enable/disable the mode
- **invert_switch** (`bool`) – invert switch state (e.g. *OFF* -> enabled, default is `False`)
- **initial_value** – initial value of the mode
- **logger** (`Optional[Logger]`) – logger to log events
- **auto_disable_after** – see variables
- **auto_disable_func** – see variables
- **calc_value_func** – see variables

`cancel()`
Remove the mode from the parent `MultiModeItem` and stop processing it

`property enabled: bool`
Returns if the value is enabled

Return type `bool`

`set_value(value, only_on_change=False)`
Set new value and recalculate overall value. If `enable_on_value` is set, setting a value will also enable the mode.

Parameters

- **value** – new value
- **only_on_change** (`bool`) – will set/enable the mode only if value differs or the mode is disabled

`property value`
Returns the current value

CHAPTER

FOURTEEN

ADDITIONAL RULE EXAMPLES

14.1 Using the scheduler

```
from datetime import time, timedelta, datetime
from HABApp import Rule


class MyRule(Rule):

    def __init__(self):
        super().__init__()

        self.run.on_day_of_week(time=time(14, 34, 20), weekdays=['Mo'], callback=self.
↪run_mondays)

        self.run.every(timedelta(seconds=5), 3, self.run_every_3s, 'arg 1', asdf='kwarg 1
↪')

        self.run.on_workdays(time(15, 00), self.run_workdays)
        self.run.on_weekends(time(15, 00), self.run_weekends)

    def run_every_3s(self, arg, asdf = None):
        print(f'run_ever_3s: {datetime.now().replace(microsecond=0)} : {arg}, {asdf}')

    def run_mondays(self):
        print('Today is monday!')

    def run_workdays(self):
        print('Today is a workday!')

    def run_weekends(self):
        print('Finally weekend!')


MyRule()
```

14.2 Mirror openHAB events to a MQTT Broker

```
import HABApp
from HABApp.openhab.events import ItemStateEvent
from HABApp.openhab.items import Thing
from HABApp.mqtt.items import MqttItem


class ExampleOpenhabToMQTTRule(HABApp.Rule):
    """This Rule mirrors all updates from OpenHAB to MQTT"""

    def __init__(self):
        super().__init__()

        for item in HABApp.core.Items.get_all_items():
            if isinstance(item, (Thing, MqttItem)):
                continue
            item.listen_event(self.process_update, ItemStateEvent)

    def process_update(self, event):
        assert isinstance(event, ItemStateEvent)

        print(f'/openhab/{event.name} <- {event.value}')
        self.mqtt.publish(f'/openhab/{event.name}', str(event.value))


ExampleOpenhabToMQTTRule()
```

14.3 Trigger an event when an item is constant

Get an even when the item is constant for 5 and for 10 seconds.

```
import HABApp
from HABApp.core.items import Item
from HABApp.core.events import ItemNoChangeEvent

class MyRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        my_item = Item.get_item('test_watch')

        my_item.watch_change(5)     # Create event when item doesn't change for  5 secs
        my_item.watch_change(10)    # Create event when item doesn't change for 10 secs

        # Listen to these events
        self.listen_event(my_item, self.item_constant, ItemNoChangeEvent)

        # Set the item to a value
        my_item.set_value('my_value')
```

(continues on next page)

(continued from previous page)

```
    def item_constant(self, event):
        print(f'{event}')

MyRule()
```

```
<ItemNoChangeEvent name: test_watch, seconds: 5>
<ItemNoChangeEvent name: test_watch, seconds: 10>
```

14.4 Turn something off after movement

Turn a device off 30 seconds after one of the movement sensors in a room signals movement.

```
import HABApp
from HABApp.core.items import Item
from HABApp.core.events import ValueUpdateEvent

class MyCountdownRule(HABApp.Rule):
    def __init__(self):
        super().__init__()

        self.countdown = self.run.countdown(30, self.switch_off)
        self.device = Item.get_item('my_device')

        self.movement1 = Item.get_item('movement_sensor1')
        self.movement1.listen_event(self.movement, ValueUpdateEvent)

        self.movement2 = Item.get_item('movement_sensor2')
        self.movement2.listen_event(self.movement, ValueUpdateEvent)

    def movement(self, event: ValueUpdateEvent):
        if self.device != 'ON':
            self.device.post_value('ON')

        self.countdown.reset()

    def switch_off(self):
        self.device.post_value('OFF')

MyCountdownRule()
```

14.5 Process Errors in Rules

This example shows how to create a rule with a function which will be called when **any** rule throws an error. The rule function then can push the error message to an openhab item or e.g. use Pushover to send the error message to the mobile device (see *Avanced Usage* for more information).

```
import HABApp
from HABApp.core.events.habapp_events import HABAppException

class NotifyOnError(HABApp.Rule):
    def __init__(self):
        super().__init__()

        # Listen to all errors
        self.listen_event('HABApp.Errors', self.on_error, HABAppException)

    def on_error(self, error_event: HABAppException):
        msg = event.to_str() if isinstance(event, HABAppException) else event
        print(msg)

NotifyOnError()


# this is a faulty example. Do not create this part!
class FaultyRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        self.run.soon(self.faulty_function)

    def faulty_function(self):
        1 / 0
FaultyRule()
```

CHAPTER

FIFTEEN

TIPS & TRICKS

15.1 yml files

15.1.1 Entry sharing

If the values should be reused yml features anchors with & which then can be referenced with *. This allows to reuse the defined structures:

```
my_key_value_pairs:  &my_kv  # <-- this creates the anchor node with the name my_kv
  4: 99      # Light Threshold
  5: 8       # Operation Mode
  7: 20      # Customer Function

value_1: *my_kv  # <-- '*my_kv' references the anchor node my_kv
value_2: *my_kv

value_3:
  <<: *my_kv    # <-- '<<: *my_kv' references and inserts the content (!) of the anchor␣
↪node my_kv
  4: 80         #                  and then overwrites parameter 4
```

15.2 openHAB

15.2.1 autoupdate

If external devices are capable of reporting their state (e.g. Z-Wave) it is always advised to use disable `autoupdate` for these items. This prevents openhab from guessing the item state based on the command and forces it to use the actual reported value. If in doubt if the device supports reporting their state watch the state after sending a command with `autoupdate` off. If the state changes `autoupdate` can remain off.

In the `*.items` file `autoupdate` can be disabled by adding the following statement in the metadata field.

` Number MyItem { channel = "zwave:my_zwave_link", autoupdate="false" } `

It's also possible with textual thing configuration to add it as metadata.

CHAPTER

SIXTEEN

CLASS REFERENCE

Reference for returned classes from some functions. These are not intended to be created by the user.

16.1 Watches

16.1.1 ItemNoUpdateWatch

class HABApp.core.items.base_item_watch.**ItemNoUpdateWatch**(*name*, *secs*)

EVENT
alias of *HABApp.core.events.events.ItemNoUpdateEvent*

cancel()
Cancel the item watch

listen_event(*callback*)
Listen to (only) the event that is emitted by this watcher

Return type EventBusListener

16.1.2 ItemNoChangeWatch

class HABApp.core.items.base_item_watch.**ItemNoChangeWatch**(*name*, *secs*)

EVENT
alias of *HABApp.core.events.events.ItemNoChangeEvent*

cancel()
Cancel the item watch

listen_event(*callback*)
Listen to (only) the event that is emitted by this watcher

Return type EventBusListener

16.2 Scheduler

16.2.1 OneTimeJob

class eascheduler.jobs.**OneTimeJob**(*parent*, *func*)

cancel()
Cancel the job.

get_next_run()
Return the next execution timestamp.

Return type datetime

16.2.2 CountdownJob

class eascheduler.jobs.**CountdownJob**(*parent*, *func*)

cancel()
Cancel the job.

countdown(*time*)
Set the time after which the job will be executed.

Parameters **time** (Union[timedelta, float, int]) – time

Return type *CountdownJob*

get_next_run()
Return the next execution timestamp.

Return type datetime

16.2.3 ReoccurringJob

class eascheduler.jobs.**ReoccurringJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

interval(*interval*)
Set the interval at which the task will run.

Parameters **interval** (Union[int, float, timedelta]) – interval in secs or a timedelta obj

Return type *ReoccurringJob*

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

16.2.4 DayOfWeekJob

class eascheduler.jobs.**DayOfWeekJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

time(*time*)
Set a time of day when the job will run.

Parameters **time** (Union[time, datetime]) – time

Return type *DayOfWeekJob*

weekdays(*weekdays*)
Set the weekdays when the job will run.

Parameters **weekdays** (Union[str, Iterable[Union[str, int]]]) – Day group names (e.g. 'all', 'weekend', 'workdays'), an iterable with day names (e.g. ['Mon', 'Fri']) or an iterable with the isoweekday values (e.g. [1, 5]).

Return type *DayOfWeekJob*

16.2.5 DawnJob

class eascheduler.jobs.**DawnJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

16.2.6 SunriseJob

class eascheduler.jobs.**SunriseJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

16.2.7 SunsetJob

class eascheduler.jobs.**SunsetJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

16.2.8 DuskJob

class eascheduler.jobs.**DuskJob**(*parent*, *func*)

boundary_func(*func*)
Add a function which will be called when the datetime changes. Use this to implement custom boundaries. Use None to disable the boundary function.

Parameters **func** (Optional[Callable[[datetime], datetime]]) – Function which returns a datetime obj, arg is a datetime with the next run time. Return SKIP_EXECUTION together with a reoccurring job to skip the proposed run time.

Return type DateTimeJobBase

cancel()
Cancel the job.

earliest(*time_obj*)
Set earliest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run earlier

Return type DateTimeJobBase

get_next_run()
Return the next execution timestamp.

Return type datetime

jitter(*start*, *stop=None*)
Add a random jitter per call in the interval [start <= secs <= stop] to the next run. If stop is omitted start must be positive and the interval will be [-start <= secs <= start] Passing None as start will disable jitter.

Parameters

- **start** (Union[int, float, None]) – Interval start or None to disable jitter
- **stop** (Union[int, float, None]) – Interval stop or None to build interval based on start

Return type DateTimeJobBase

latest(*time_obj*)
Set latest boundary as time of day. None will disable boundary.

Parameters **time_obj** (Optional[time]) – time obj, scheduler will not run later

Return type DateTimeJobBase

offset(*timedelta_obj*)
Set a constant offset to the calculation of the next run. None will disable the offset.

Parameters **timedelta_obj** (Optional[timedelta]) – constant offset

Return type DateTimeJobBase

CHAPTER

SEVENTEEN

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

h

HABApp.openhab.interface, 48
HABApp.util, 111

INDEX

Symbols

__init__() (*HABApp.util.CounterItem method*), 115

A

add_mode() (*HABApp.util.multimode.MultiModeItem method*), 121
add_value() (*HABApp.util.Statistics method*), 116
aggregation_func() (*HABApp.core.items.AggregationItem method*), 40
aggregation_period() (*HABApp.core.items.AggregationItem method*), 40
aggregation_source() (*HABApp.core.items.AggregationItem method*), 40
AggregationItem (*class in HABApp.core.items*), 40
all_modes() (*HABApp.util.multimode.MultiModeItem method*), 121
AsyncHttpConnection (*class in HABApp.rule.interfaces*), 109
at() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 23

B

BaseValueItem (*class in HABApp.core.items*), 42
boundary_func() (*eascheduler.jobs.DawnJob method*), 134
boundary_func() (*eascheduler.jobs.DayOfWeekJob method*), 133
boundary_func() (*eascheduler.jobs.DuskJob method*), 137
boundary_func() (*eascheduler.jobs.ReoccurringJob method*), 132
boundary_func() (*eascheduler.jobs.SunriseJob method*), 135
boundary_func() (*eascheduler.jobs.SunsetJob method*), 136

C

calculate_value() (*HABApp.util.multimode.MultiModeItem method*), 121
cancel() (*eascheduler.jobs.CountdownJob method*), 132
cancel() (*eascheduler.jobs.DawnJob method*), 134
cancel() (*eascheduler.jobs.DayOfWeekJob method*), 133
cancel() (*eascheduler.jobs.DuskJob method*), 137
cancel() (*eascheduler.jobs.OneTimeJob method*), 132
cancel() (*eascheduler.jobs.ReoccurringJob method*), 132
cancel() (*eascheduler.jobs.SunriseJob method*), 135
cancel() (*eascheduler.jobs.SunsetJob method*), 136
cancel() (*HABApp.core.items.base_item_watch.ItemNoChangeWatch method*), 131
cancel() (*HABApp.core.items.base_item_watch.ItemNoUpdateWatch method*), 131
cancel() (*HABApp.util.multimode.SwitchItemValueMode method*), 123
cancel() (*HABApp.util.multimode.ValueMode method*), 122
channel_link_exists() (*in module HABApp.openhab.interface*), 50
ChannelTriggeredEvent (*class in HABApp.openhab.events*), 80
ColorItem (*class in HABApp.core.items*), 37
ColorItem (*class in HABApp.openhab.items*), 62
ContactItem (*class in HABApp.openhab.items*), 53
countdown() (*eascheduler.jobs.CountdownJob method*), 132
countdown() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 24
CountdownJob (*class in eascheduler.jobs*), 132
CounterItem (*class in HABApp.util*), 115
create_channel_link() (*in module HABApp.openhab.interface*), 50
create_item() (*in module HABApp.openhab.interface*), 49

D

DawnJob (*class in eascheduler.jobs*), 134
DayOfWeekJob (*class in eascheduler.jobs*), 133
decrease() (*HABApp.util.CounterItem method*), 115
DictParameter (*class in HABApp.parameters*), 33

DimmerItem (*class in HABApp.openhab.items*), 58
down() (*HABApp.openhab.items.RollershutterItem method*), 60
DuskJob (*class in eascheduler.jobs*), 137

E

earliest() (*eascheduler.jobs.DawnJob method*), 134
earliest() (*eascheduler.jobs.DayOfWeekJob method*), 133
earliest() (*eascheduler.jobs.DuskJob method*), 137
earliest() (*eascheduler.jobs.ReoccurringJob method*), 132
earliest() (*eascheduler.jobs.SunriseJob method*), 135
earliest() (*eascheduler.jobs.SunsetJob method*), 136
enabled (*HABApp.util.multimode.SwitchItemValueMode property*), 123
enabled (*HABApp.util.multimode.ValueMode property*), 122
EVENT (*HABApp.core.items.base_item_watch.ItemNoChangeWatch attribute*), 131
EVENT (*HABApp.core.items.base_item_watch.ItemNoUpdateWatch attribute*), 131
EventFilter (*class in HABApp.core.events*), 22
every() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 24
every_hour() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 26
every_minute() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 26
execute_subprocess() (*HABApp.Rule method*), 29

F

FinishedProcessInfo (*class in HABApp.rule*), 27

G

get() (*HABApp.rule.interfaces.AsyncHttpConnection method*), 109
get_channel_link() (*in module HABApp.openhab.interface*), 50
get_client_session() (*HABApp.rule.interfaces.AsyncHttpConnection method*), 110
get_create_item() (*HABApp.core.items.AggregationItem class method*), 40
get_create_item() (*HABApp.core.items.Item class method*), 35
get_create_item() (*HABApp.mqtt.items.MqttItem class method*), 98
get_create_item() (*HABApp.mqtt.items.MqttPairItem class method*), 100
get_create_item() (*HABApp.util.multimode.MultiModeItem class method*), 120
get_item() (*HABApp.core.items.AggregationItem class method*), 40
get_item() (*HABApp.core.items.BaseValueItem class method*), 42
get_item() (*HABApp.core.items.ColorItem class method*), 37
get_item() (*HABApp.core.items.Item class method*), 35
get_item() (*HABApp.mqtt.items.MqttItem class method*), 98
get_item() (*HABApp.mqtt.items.MqttPairItem class method*), 101
get_item() (*HABApp.openhab.items.ColorItem class method*), 62
get_item() (*HABApp.openhab.items.ContactItem class method*), 53
get_item() (*HABApp.openhab.items.DimmerItem class method*), 58
get_item() (*HABApp.openhab.items.GroupItem class method*), 71
get_item() (*HABApp.openhab.items.ImageItem class method*), 73
get_item() (*HABApp.openhab.items.LocationItem class method*), 67
get_item() (*HABApp.openhab.items.NumberItem class method*), 51
get_item() (*HABApp.openhab.items.PlayerItem class method*), 69
get_item() (*HABApp.openhab.items.RollershutterItem class method*), 60
get_item() (*HABApp.openhab.items.StringItem class method*), 65
get_item() (*HABApp.openhab.items.SwitchItem class method*), 55
get_item() (*HABApp.openhab.items.Thing class method*), 75
get_item() (*in module HABApp.openhab.interface*), 48
get_mode() (*HABApp.util.multimode.MultiModeItem method*), 121
get_next_run() (*eascheduler.jobs.CountdownJob method*), 132
get_next_run() (*eascheduler.jobs.DawnJob method*), 135
get_next_run() (*eascheduler.jobs.DayOfWeekJob method*), 133
get_next_run() (*eascheduler.jobs.DuskJob method*), 137
get_next_run() (*eascheduler.jobs.OneTimeJob method*), 132
get_next_run() (*eascheduler.jobs.ReoccurringJob method*), 132
get_next_run() (*eascheduler.jobs.SunriseJob method*), 135
get_next_run() (*eascheduler.jobs.SunsetJob method*), 136
get_persistence_data() (*HABApp.openhab.items.ColorItem method*),

63
get_persistence_data() (*HABApp.openhab.items.ContactItem method*), 53
get_persistence_data() (*HABApp.openhab.items.DimmerItem method*), 58
get_persistence_data() (*HABApp.openhab.items.GroupItem method*), 71
get_persistence_data() (*HABApp.openhab.items.ImageItem method*), 73
get_persistence_data() (*HABApp.openhab.items.LocationItem method*), 67
get_persistence_data() (*HABApp.openhab.items.NumberItem method*), 51
get_persistence_data() (*HABApp.openhab.items.PlayerItem method*), 69
get_persistence_data() (*HABApp.openhab.items.RollershutterItem method*), 60
get_persistence_data() (*HABApp.openhab.items.StringItem method*), 65
get_persistence_data() (*HABApp.openhab.items.SwitchItem method*), 55
get_persistence_data() (*in module HABApp.openhab.interface*), 49
get_rgb() (*HABApp.core.items.ColorItem method*), 37
get_rgb() (*HABApp.openhab.items.ColorItem method*), 63
get_thing() (*in module HABApp.openhab.interface*), 49
get_value() (*HABApp.core.items.AggregationItem method*), 40
get_value() (*HABApp.core.items.BaseValueItem method*), 42
get_value() (*HABApp.core.items.ColorItem method*), 37
get_value() (*HABApp.core.items.Item method*), 35
get_value() (*HABApp.mqtt.items.MqttItem method*), 98
get_value() (*HABApp.mqtt.items.MqttPairItem method*), 101
get_value() (*HABApp.openhab.items.ColorItem method*), 63
get_value() (*HABApp.openhab.items.ContactItem method*), 54
get_value() (*HABApp.openhab.items.DimmerItem method*), 58
get_value() (*HABApp.openhab.items.GroupItem method*), 71
get_value() (*HABApp.openhab.items.ImageItem method*), 73
get_value() (*HABApp.openhab.items.LocationItem method*), 68
get_value() (*HABApp.openhab.items.NumberItem method*), 52
get_value() (*HABApp.openhab.items.PlayerItem method*), 70
get_value() (*HABApp.openhab.items.RollershutterItem method*), 60
get_value() (*HABApp.openhab.items.StringItem method*), 66
get_value() (*HABApp.openhab.items.SwitchItem method*), 56
GroupItem (*class in HABApp.openhab.items*), 71
GroupItemStateChangedEvent (*class in HABApp.openhab.events*), 79

H

HABApp.openhab.interface
    module, 48
HABApp.util
    module, 111
HABAppException (*class in HABApp.core.events.habapp_events*), 105
HABAppScheduler (*class in HABApp.rule.habappscheduler*), 23
hsb_to_rgb() (*in module HABApp.util.functions*), 114

I

ImageItem (*class in HABApp.openhab.items*), 73
increase() (*HABApp.util.CounterItem method*), 115
interval() (*eascheduler.jobs.ReoccurringJob method*), 133
is_closed() (*HABApp.openhab.items.ContactItem method*), 54
is_down() (*HABApp.openhab.items.RollershutterItem method*), 61
is_off() (*HABApp.core.items.ColorItem method*), 37
is_off() (*HABApp.openhab.items.ColorItem method*), 63
is_off() (*HABApp.openhab.items.DimmerItem method*), 58
is_off() (*HABApp.openhab.items.SwitchItem method*), 56
is_on() (*HABApp.core.items.ColorItem method*), 37
is_on() (*HABApp.openhab.items.ColorItem method*), 63
is_on() (*HABApp.openhab.items.DimmerItem method*), 58

is_on() (*HABApp.openhab.items.SwitchItem method*), 56
is_open() (*HABApp.openhab.items.ContactItem method*), 54
is_up() (*HABApp.openhab.items.RollershutterItem method*), 61
Item (*class in HABApp.core.items*), 35
item_exists() (*in module HABApp.openhab.interface*), 49
ItemAddedEvent (*class in HABApp.openhab.events*), 78
ItemCommandEvent (*class in HABApp.openhab.events*), 78
ItemNoChangeEvent (*class in HABApp.core.events*), 45
ItemNoChangeWatch (*class in HABApp.core.items.base_item_watch*), 131
ItemNoUpdateEvent (*class in HABApp.core.events*), 44
ItemNoUpdateWatch (*class in HABApp.core.items.base_item_watch*), 131
ItemRemovedEvent (*class in HABApp.openhab.events*), 79
ItemStateChangedEvent (*class in HABApp.openhab.events*), 77
ItemStateChangedEventFilter (*class in HABApp.openhab.events*), 82
ItemStateEvent (*class in HABApp.openhab.events*), 77
ItemStateEventFilter (*class in HABApp.openhab.events*), 82
ItemStatePredictedEvent (*class in HABApp.openhab.events*), 79
ItemUpdatedEvent (*class in HABApp.openhab.events*), 78

J

jitter() (*eascheduler.jobs.DawnJob method*), 135
jitter() (*eascheduler.jobs.DayOfWeekJob method*), 134
jitter() (*eascheduler.jobs.DuskJob method*), 137
jitter() (*eascheduler.jobs.ReoccurringJob method*), 133
jitter() (*eascheduler.jobs.SunriseJob method*), 136
jitter() (*eascheduler.jobs.SunsetJob method*), 136

L

last_change (*HABApp.core.items.AggregationItem property*), 41
last_change (*HABApp.core.items.BaseValueItem property*), 43
last_change (*HABApp.core.items.ColorItem property*), 39
last_change (*HABApp.core.items.Item property*), 36
last_change (*HABApp.mqtt.items.MqttItem property*), 100
last_change (*HABApp.mqtt.items.MqttPairItem property*), 102
last_change (*HABApp.openhab.items.ColorItem property*), 65
last_change (*HABApp.openhab.items.ContactItem property*), 55
last_change (*HABApp.openhab.items.DimmerItem property*), 60
last_change (*HABApp.openhab.items.GroupItem property*), 73
last_change (*HABApp.openhab.items.ImageItem property*), 75
last_change (*HABApp.openhab.items.LocationItem property*), 69
last_change (*HABApp.openhab.items.NumberItem property*), 53
last_change (*HABApp.openhab.items.PlayerItem property*), 71
last_change (*HABApp.openhab.items.RollershutterItem property*), 62
last_change (*HABApp.openhab.items.StringItem property*), 67
last_change (*HABApp.openhab.items.SwitchItem property*), 57
last_change (*HABApp.openhab.items.Thing property*), 76
last_update (*HABApp.core.items.AggregationItem property*), 41
last_update (*HABApp.core.items.BaseValueItem property*), 43
last_update (*HABApp.core.items.ColorItem property*), 39
last_update (*HABApp.core.items.Item property*), 36
last_update (*HABApp.mqtt.items.MqttItem property*), 100
last_update (*HABApp.mqtt.items.MqttPairItem property*), 102
last_update (*HABApp.openhab.items.ColorItem property*), 65
last_update (*HABApp.openhab.items.ContactItem property*), 55
last_update (*HABApp.openhab.items.DimmerItem property*), 60
last_update (*HABApp.openhab.items.GroupItem property*), 73
last_update (*HABApp.openhab.items.ImageItem property*), 75
last_update (*HABApp.openhab.items.LocationItem property*), 69
last_update (*HABApp.openhab.items.NumberItem property*), 53
last_update (*HABApp.openhab.items.PlayerItem property*), 71
last_update (*HABApp.openhab.items.RollershutterItem property*), 62
last_update (*HABApp.openhab.items.StringItem prop-*

erty), 67
last_update (*HABApp.openhab.items.SwitchItem property*), 57
last_update (*HABApp.openhab.items.Thing property*), 76
latest() (*eascheduler.jobs.DawnJob method*), 135
latest() (*eascheduler.jobs.DayOfWeekJob method*), 134
latest() (*eascheduler.jobs.DuskJob method*), 137
latest() (*eascheduler.jobs.ReoccurringJob method*), 133
latest() (*eascheduler.jobs.SunriseJob method*), 136
latest() (*eascheduler.jobs.SunsetJob method*), 137
listen_event() (*HABApp.core.items.AggregationItem method*), 40
listen_event() (*HABApp.core.items.base_item_watch.ItemNoChangeWatch method*), 131
listen_event() (*HABApp.core.items.base_item_watch.ItemNoUpdateWatch method*), 131
listen_event() (*HABApp.core.items.BaseValueItem method*), 42
listen_event() (*HABApp.core.items.ColorItem method*), 37
listen_event() (*HABApp.core.items.Item method*), 35
listen_event() (*HABApp.mqtt.items.MqttItem method*), 99
listen_event() (*HABApp.mqtt.items.MqttPairItem method*), 101
listen_event() (*HABApp.openhab.items.ColorItem method*), 63
listen_event() (*HABApp.openhab.items.ContactItem method*), 54
listen_event() (*HABApp.openhab.items.DimmerItem method*), 58
listen_event() (*HABApp.openhab.items.GroupItem method*), 72
listen_event() (*HABApp.openhab.items.ImageItem method*), 74
listen_event() (*HABApp.openhab.items.LocationItem method*), 68
listen_event() (*HABApp.openhab.items.NumberItem method*), 52
listen_event() (*HABApp.openhab.items.PlayerItem method*), 70
listen_event() (*HABApp.openhab.items.RollershutterItem method*), 61
listen_event() (*HABApp.openhab.items.StringItem method*), 66
listen_event() (*HABApp.openhab.items.SwitchItem method*), 56
listen_event() (*HABApp.openhab.items.Thing method*), 75
listen_event() (*HABApp.Rule method*), 28
LocationItem (*class in HABApp.openhab.items*), 67

M

max() (*in module HABApp.util.functions*), 113
min() (*in module HABApp.util.functions*), 113
module
    HABApp.openhab.interface, 48
    HABApp.util, 111
mqtt (*built-in class*), 97
MqttItem (*class in HABApp.mqtt.items*), 98
MqttPairItem (*class in HABApp.mqtt.items*), 100
MqttValueChangeEvent (*class in HABApp.mqtt.events*), 103
MqttValueUpdateEvent (*class in HABApp.mqtt.events*), 102
MultiModeItem (*class in HABApp.util.multimode*), 120

N

name (*HABApp.core.items.AggregationItem property*), 41
name (*HABApp.core.items.BaseValueItem property*), 43
name (*HABApp.core.items.ColorItem property*), 39
name (*HABApp.core.items.Item property*), 36
name (*HABApp.mqtt.items.MqttItem property*), 100
name (*HABApp.mqtt.items.MqttPairItem property*), 102
name (*HABApp.openhab.items.ColorItem property*), 65
name (*HABApp.openhab.items.ContactItem property*), 55
name (*HABApp.openhab.items.DimmerItem property*), 60
name (*HABApp.openhab.items.GroupItem property*), 73
name (*HABApp.openhab.items.ImageItem property*), 75
name (*HABApp.openhab.items.LocationItem property*), 69
name (*HABApp.openhab.items.NumberItem property*), 53
name (*HABApp.openhab.items.PlayerItem property*), 71
name (*HABApp.openhab.items.RollershutterItem property*), 62
name (*HABApp.openhab.items.StringItem property*), 67
name (*HABApp.openhab.items.SwitchItem property*), 57
name (*HABApp.openhab.items.Thing property*), 76
NumberItem (*class in HABApp.openhab.items*), 51

O

off() (*HABApp.openhab.items.ColorItem method*), 63
off() (*HABApp.openhab.items.DimmerItem method*), 59
off() (*HABApp.openhab.items.SwitchItem method*), 56
offset() (*eascheduler.jobs.DawnJob method*), 135
offset() (*eascheduler.jobs.DayOfWeekJob method*), 134
offset() (*eascheduler.jobs.DuskJob method*), 138
offset() (*eascheduler.jobs.ReoccurringJob method*), 133
offset() (*eascheduler.jobs.SunriseJob method*), 136
offset() (*eascheduler.jobs.SunsetJob method*), 137
oh_post_update() (*HABApp.openhab.items.ColorItem method*), 63
oh_post_update() (*HABApp.openhab.items.ContactItem method*), 54

oh_post_update() (*HABApp.openhab.items.DimmerItem method*), 59
oh_post_update() (*HABApp.openhab.items.GroupItem method*), 72
oh_post_update() (*HABApp.openhab.items.ImageItem method*), 74
oh_post_update() (*HABApp.openhab.items.LocationItem method*), 68
oh_post_update() (*HABApp.openhab.items.NumberItem method*), 52
oh_post_update() (*HABApp.openhab.items.PlayerItem method*), 70
oh_post_update() (*HABApp.openhab.items.RollershutterItem method*), 61
oh_post_update() (*HABApp.openhab.items.StringItem method*), 66
oh_post_update() (*HABApp.openhab.items.SwitchItem method*), 56
oh_send_command() (*HABApp.openhab.items.ColorItem method*), 63
oh_send_command() (*HABApp.openhab.items.ContactItem method*), 54
oh_send_command() (*HABApp.openhab.items.DimmerItem method*), 59
oh_send_command() (*HABApp.openhab.items.GroupItem method*), 72
oh_send_command() (*HABApp.openhab.items.ImageItem method*), 74
oh_send_command() (*HABApp.openhab.items.LocationItem method*), 68
oh_send_command() (*HABApp.openhab.items.NumberItem method*), 52
oh_send_command() (*HABApp.openhab.items.PlayerItem method*), 70
oh_send_command() (*HABApp.openhab.items.RollershutterItem method*), 61
oh_send_command() (*HABApp.openhab.items.StringItem method*), 66
oh_send_command() (*HABApp.openhab.items.SwitchItem method*), 56
on() (*HABApp.openhab.items.ColorItem method*), 64
on() (*HABApp.openhab.items.DimmerItem method*), 59
on() (*HABApp.openhab.items.SwitchItem method*), 56
on_day_of_week() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 24
on_every_day() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 24
on_sun_dawn() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 25
on_sun_dusk() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 25
on_sunrise() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 25
on_sunset() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 25
on_weekends() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 26
on_workdays() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 26
OneTimeJob (*class in eascheduler.jobs*), 132

P

Parameter (*class in HABApp.parameters*), 33
percent() (*HABApp.openhab.items.ColorItem method*), 64
percent() (*HABApp.openhab.items.DimmerItem method*), 59
percent() (*HABApp.openhab.items.RollershutterItem method*), 61
PlayerItem (*class in HABApp.openhab.items*), 69
post() (*HABApp.rule.interfaces.AsyncHttpConnection method*), 109
post_event() (*HABApp.Rule method*), 28
post_rgb() (*HABApp.core.items.ColorItem method*), 37
post_rgb() (*HABApp.openhab.items.ColorItem method*), 64
post_update() (*in module HABApp.openhab.interface*), 48
post_value() (*HABApp.core.items.AggregationItem method*), 41
post_value() (*HABApp.core.items.BaseValueItem method*), 42
post_value() (*HABApp.core.items.ColorItem method*), 38
post_value() (*HABApp.core.items.Item method*), 36
post_value() (*HABApp.mqtt.items.MqttItem method*), 99
post_value() (*HABApp.mqtt.items.MqttPairItem method*), 101
post_value() (*HABApp.openhab.items.ColorItem method*), 64
post_value() (*HABApp.openhab.items.ContactItem method*), 54
post_value() (*HABApp.openhab.items.DimmerItem method*), 59
post_value() (*HABApp.openhab.items.GroupItem method*), 72
post_value() (*HABApp.openhab.items.ImageItem method*), 74
post_value() (*HABApp.openhab.items.LocationItem method*), 68
post_value() (*HABApp.openhab.items.NumberItem method*), 52
post_value() (*HABApp.openhab.items.PlayerItem method*), 70
post_value() (*HABApp.openhab.items.RollershutterItem method*), 61

post_value() (*HABApp.openhab.items.StringItem method*), 66
post_value() (*HABApp.openhab.items.SwitchItem method*), 56
post_value() (*HABApp.util.CounterItem method*), 115
publish() (*HABApp.mqtt.items.MqttItem method*), 99
publish() (*HABApp.mqtt.items.MqttPairItem method*), 101
publish() (*mqtt method*), 97
put() (*HABApp.rule.interfaces.AsyncHttpConnection method*), 110

R

register_cancel_obj() (*HABApp.Rule method*), 29
register_on_unload() (*HABApp.Rule method*), 29
remove_channel_link() (*in module HABApp.openhab.interface*), 50
remove_item() (*in module HABApp.openhab.interface*), 49
remove_metadata() (*in module HABApp.openhab.interface*), 49
remove_mode() (*HABApp.util.multimode.MultiModeItem method*), 120
ReoccurringJob (*class in eascheduler.jobs*), 132
RequestFileLoadEvent (*class in HABApp.core.events.habapp_events*), 105
RequestFileUnloadEvent (*class in HABApp.core.events.habapp_events*), 105
reset() (*HABApp.util.CounterItem method*), 115
rgb_to_hsb() (*in module HABApp.util.functions*), 114
RollershutterItem (*class in HABApp.openhab.items*), 60
Rule (*class in HABApp*), 28

S

send_command() (*in module HABApp.openhab.interface*), 48
set_enabled() (*HABApp.util.multimode.ValueMode method*), 122
set_file_validator() (*in module HABApp.parameters*), 32
set_metadata() (*in module HABApp.openhab.interface*), 50
set_rgb() (*HABApp.core.items.ColorItem method*), 38
set_rgb() (*HABApp.openhab.items.ColorItem method*), 64
set_value() (*HABApp.core.items.AggregationItem method*), 41
set_value() (*HABApp.core.items.BaseValueItem method*), 43
set_value() (*HABApp.core.items.ColorItem method*), 38
set_value() (*HABApp.core.items.Item method*), 36
set_value() (*HABApp.mqtt.items.MqttItem method*), 99
set_value() (*HABApp.mqtt.items.MqttPairItem method*), 101
set_value() (*HABApp.openhab.items.ColorItem method*), 64
set_value() (*HABApp.openhab.items.ContactItem method*), 54
set_value() (*HABApp.openhab.items.DimmerItem method*), 59
set_value() (*HABApp.openhab.items.GroupItem method*), 72
set_value() (*HABApp.openhab.items.ImageItem method*), 74
set_value() (*HABApp.openhab.items.LocationItem method*), 68
set_value() (*HABApp.openhab.items.NumberItem method*), 52
set_value() (*HABApp.openhab.items.PlayerItem method*), 70
set_value() (*HABApp.openhab.items.RollershutterItem method*), 61
set_value() (*HABApp.openhab.items.StringItem method*), 66
set_value() (*HABApp.openhab.items.SwitchItem method*), 57
set_value() (*HABApp.util.CounterItem method*), 115
set_value() (*HABApp.util.multimode.SwitchItemValueMode method*), 123
set_value() (*HABApp.util.multimode.ValueMode method*), 122
soon() (*HABApp.rule.habappscheduler.HABAppScheduler method*), 26
Statistics (*class in HABApp.util*), 116
StringItem (*class in HABApp.openhab.items*), 65
subscribe() (*mqtt method*), 97
SunriseJob (*class in eascheduler.jobs*), 135
SunsetJob (*class in eascheduler.jobs*), 136
SwitchItem (*class in HABApp.openhab.items*), 55
SwitchItemValueMode (*class in HABApp.util.multimode*), 122

T

Thing (*class in HABApp.openhab.items*), 75
ThingConfigStatusInfoEvent (*class in HABApp.openhab.events*), 81
ThingFirmwareStatusInfoEvent (*class in HABApp.openhab.events*), 82
ThingStatusInfoChangedEvent (*class in HABApp.openhab.events*), 80
ThingStatusInfoEvent (*class in HABApp.openhab.events*), 81
time() (*eascheduler.jobs.DayOfWeekJob method*), 134

to_str() (*HABApp.core.events.habapp_events.HABAppException method*), 105

U

unsubscribe() (*mqtt method*), 97
up() (*HABApp.openhab.items.RollershutterItem method*), 61
update() (*HABApp.util.Statistics method*), 116

V

value (*HABApp.parameters.DictParameter property*), 34
value (*HABApp.parameters.Parameter property*), 33
value (*HABApp.util.multimode.SwitchItemValueMode property*), 123
value (*HABApp.util.multimode.ValueMode property*), 122
ValueChangeEvent (*class in HABApp.core.events*), 44
ValueChangeEventFilter (*class in HABApp.core.events*), 22
ValueMode (*class in HABApp.util.multimode*), 121
ValueUpdateEvent (*class in HABApp.core.events*), 44
ValueUpdateEventFilter (*class in HABApp.core.events*), 22

W

watch_change() (*HABApp.core.items.AggregationItem method*), 41
watch_change() (*HABApp.core.items.BaseValueItem method*), 43
watch_change() (*HABApp.core.items.ColorItem method*), 38
watch_change() (*HABApp.core.items.Item method*), 36
watch_change() (*HABApp.mqtt.items.MqttItem method*), 99
watch_change() (*HABApp.mqtt.items.MqttPairItem method*), 102
watch_change() (*HABApp.openhab.items.ColorItem method*), 65
watch_change() (*HABApp.openhab.items.ContactItem method*), 54
watch_change() (*HABApp.openhab.items.DimmerItem method*), 59
watch_change() (*HABApp.openhab.items.GroupItem method*), 72
watch_change() (*HABApp.openhab.items.ImageItem method*), 74
watch_change() (*HABApp.openhab.items.LocationItem method*), 68
watch_change() (*HABApp.openhab.items.NumberItem method*), 52
watch_change() (*HABApp.openhab.items.PlayerItem method*), 70
watch_change() (*HABApp.openhab.items.RollershutterItem method*), 61
watch_change() (*HABApp.openhab.items.StringItem method*), 66
watch_change() (*HABApp.openhab.items.SwitchItem method*), 57
watch_change() (*HABApp.openhab.items.Thing method*), 76
watch_update() (*HABApp.core.items.AggregationItem method*), 41
watch_update() (*HABApp.core.items.BaseValueItem method*), 43
watch_update() (*HABApp.core.items.ColorItem method*), 39
watch_update() (*HABApp.core.items.Item method*), 36
watch_update() (*HABApp.mqtt.items.MqttItem method*), 99
watch_update() (*HABApp.mqtt.items.MqttPairItem method*), 102
watch_update() (*HABApp.openhab.items.ColorItem method*), 65
watch_update() (*HABApp.openhab.items.ContactItem method*), 55
watch_update() (*HABApp.openhab.items.DimmerItem method*), 59
watch_update() (*HABApp.openhab.items.GroupItem method*), 72
watch_update() (*HABApp.openhab.items.ImageItem method*), 75
watch_update() (*HABApp.openhab.items.LocationItem method*), 69
watch_update() (*HABApp.openhab.items.NumberItem method*), 53
watch_update() (*HABApp.openhab.items.PlayerItem method*), 71
watch_update() (*HABApp.openhab.items.RollershutterItem method*), 62
watch_update() (*HABApp.openhab.items.StringItem method*), 67
watch_update() (*HABApp.openhab.items.SwitchItem method*), 57
watch_update() (*HABApp.openhab.items.Thing method*), 76
weekdays() (*eascheduler.jobs.DayOfWeekJob method*), 134