
HABApp Documentation

Release beta

spacemanspiff2007

Apr 12, 2024

USER DOCUMENTATION

1	Installation & Usage	1
1.1	Virtual environment	1
1.2	Docker	4
1.3	Upgrading to a newer version of HABApp	6
1.4	Command line arguments	6
1.5	Usage with PyCharm	7
1.6	Install a development version of HABApp	9
2	About HABApp	11
2.1	About	11
2.2	HABApp architecture	12
2.3	HABApp folder structure	13
2.4	Integration with openHAB	13
2.5	Integration with MQTT	14
3	Configuration	15
3.1	Description	15
3.2	Example	15
3.3	Configuration Reference	17
4	Getting Started	23
4.1	First rule	23
4.2	A more generic rule	24
4.3	Interacting with items	24
4.4	Watch items for events	26
4.5	Trigger an event when an item is constant	27
4.6	Convenience functions	28
5	Logging	29
5.1	Configuration	29
5.2	Provided loggers	29
5.3	Example	29
5.4	Custom log levels	32
5.5	Logging to stdout	32
5.6	Add custom filters to loggers	32
6	Rule	33
6.1	Interacting with items	33
6.2	Interacting with events	34
6.3	Scheduler	36
6.4	Other tools and scripts	40

6.5	How to properly use rules from other rule files	42
6.6	All available functions	43
7	Parameters	47
7.1	Parameters	47
7.2	Validation	48
7.3	Create rules from Parameters	49
7.4	Parameter classes	49
8	HABApp	51
8.1	Datatypes	51
8.2	Items	54
8.3	Events	67
9	openHAB	71
9.1	Additional configuration	71
9.2	openHAB item types	72
9.3	Interaction with a openHAB	128
9.4	openHAB event types	131
9.5	Transformations	139
9.6	Textual thing configuration	139
9.7	Example openHAB rules	149
10	MQTT	153
10.1	Interaction with the MQTT broker	153
10.2	Rule Interface	153
10.3	Mqtt item types	154
10.4	Mqtt event types	161
10.5	Example MQTT rule	161
11	Advanced Usage	163
11.1	HABApp Topics	163
11.2	File properties	164
11.3	Running Python code on startup	164
11.4	Invoking openHAB actions	165
11.5	Mocking openHAB items and events for tests	166
12	asyncio	169
12.1	async http	169
13	util - helpers and utilities	173
13.1	Functions	173
13.2	Rate limiter	175
13.3	Statistics	179
13.4	Fade	180
13.5	EventListenerGroup	182
13.6	MultiModeItem	184
14	Additional rule examples	193
14.1	Using the scheduler	193
14.2	Mirror openHAB events to a MQTT Broker	194
14.3	Trigger an event when an item is constant	194
14.4	Turn something off after movement	195
14.5	Process Errors in Rules	196

15 Tips & Tricks	199
15.1 yml files	199
15.2 openHAB	199
16 Troubleshooting	201
16.1 Warnings	201
16.2 Errors	202
17 Class reference	203
17.1 Watches	203
17.2 Scheduler	204
18 Indices and tables	215
Python Module Index	217
Index	219

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 and setuptools:

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

6. Install HABApp:

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

7. Run HABApp:

```
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 Upgrade to the latest version

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 Installation of a certain version

Installing a certain version of HABApp requires the same steps used for installation or upgrading HABApp. However the final `python3 -m pip install` command is now different and contains the version number:

```
python3 -m pip install HABApp==23.12.0
```

The complete list of available versions can be found on [pypi](https://pypi.org/project/HABApp/).

1.1.4 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
```

1.1.5 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\
```

(continues on next page)

(continued from previous page)

```

↪ 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/
-----

```

1.1.6 Error message while installing ruamel.yaml

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

Run the following command to fix it:

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

1.2 Docker

1.2.1 Image installation

Installation through [docker](#) is available:

```
docker pull spacemanspiff2007/habapp:latest
```

The image supports the following environment variables.

Variable	Description
TZ	Timezone used for the container (e.g. Europe/Berlin).
USER_ID	User id at which HABApp will run (Optional, default: 9001)
GROUP_ID	Group id at which HABApp will run (Optional, default: USER_ID)
HABAPP_HOME	Directory in which the config resides (in subdirectory "config") default: habapp)

1.2.2 Running image from command line

```

docker run --rm -it --name habapp \
  -v ${PWD}/habapp_config:/habapp/config \
  -e TZ=Europe/Berlin \
  -e USER_ID=9001 \
  -e GROUP_ID=9001 \
  spacemanspiff2007/habapp:latest

```

Parameters explained

Parameter	Description
<code>--rm</code>	Remove container when stopped
<code>-it</code>	Run in interactive mode (Optional) -> You can stop HABApp by pressing STRG+C and see stdout
<code>--name habapp</code>	Give the container an unique name to interact with it
<code>-e TZ=Europe/Berlin</code>	Set environment variable with timezone
<code>-e USER_ID=9001</code>	Set environment variable with wser id at which HABApp will run (Optional, default: 9001)
<code>-e GROUP_ID=9001</code>	Set environment variable with group id at which HABApp will run (Optional, default: USER_ID)
<code>spacemanspiff2007/habapp:latest</code>	Name of the image that will be run

1.2.3 Updating image from command line

```
docker stop habapp
```

```
docker pull spacemanspiff2007/habapp:latest
```

1.2.4 Updating image 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.

1.2.5 Additional python libraries

If you want to use some additional python libraries you can do this by writing your own Dockerfile using this image as base image. The HABApp image is based on the python-slim image so you can install packages by using apt and pip.

Example Dockerfile installing scipy, pandas and numpy libraries:

```
FROM spacemanspiff2007/habapp:latest as buildimage

RUN set -eux; \
# Install required build dependencies (Optional)
  apt-get update; \
  DEBIAN_FRONTEND=noninteractive apt-get install --no-install-recommends -y \
    build-essential; \
# Prepare python packages
  pip3 wheel \
    --wheel-dir=/root/wheels \
    # Replace 'scipy pandas numpy' with your libraries
    scipy pandas numpy

FROM spacemanspiff2007/habapp:latest
```

(continues on next page)

(continued from previous page)

```

COPY --from=buildimage /root/wheels /root/wheels

RUN set -eux; \
# Install required runtime dependencies (Optional)
    apt-get update; \
    DEBIAN_FRONTEND=noninteractive apt-get install --no-install-recommends -y \
        bash; \
    apt-get clean; \
    rm -rf /var/lib/apt/lists/*; \
# Install python packages and cleanup
    pip3 install \
        --no-index \
        --find-links=/root/wheels \
        # Replace 'scipy pandas numpy' with your libraries
        scipy pandas numpy; \
    rm -rf /root/wheels

```

Build image

```
docker build -t my_habapp_extended:latest .
```

Start image (same as with provided image but the image name is different).

```

docker run --rm -it --name habapp \
    -v ${PWD}/habapp_config:/habapp/config \
    -e TZ=Europe/Berlin \
    -e USER_ID=9001 \
    -e GROUP_ID=9001 \
    my_habapp_extended:latest

```

1.3 Upgrading to a newer version of HABApp

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 Command line arguments

Execute `habapp` with “-h” to view possible command line arguments

```
habapp -h
```

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

Start HABApp

options:

(continues on next page)

(continued from previous page)

```

-h, --help          show this help message and exit
-c CONFIG, --config CONFIG
                    Path to configuration folder (where the config.yml is
                    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
-di, --debug-info   Print debug information

```

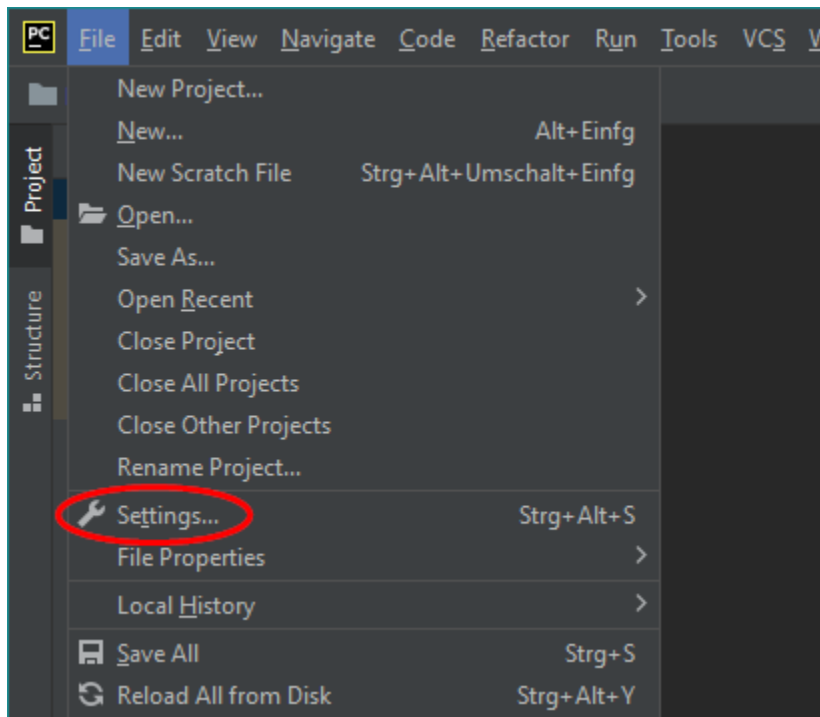
1.5 Usage with PyCharm

It's recommended to use PyCharm as an IDE for writing rules. The IDE can provide auto complete and static checks which will help write error free rules and vastly speed up development.

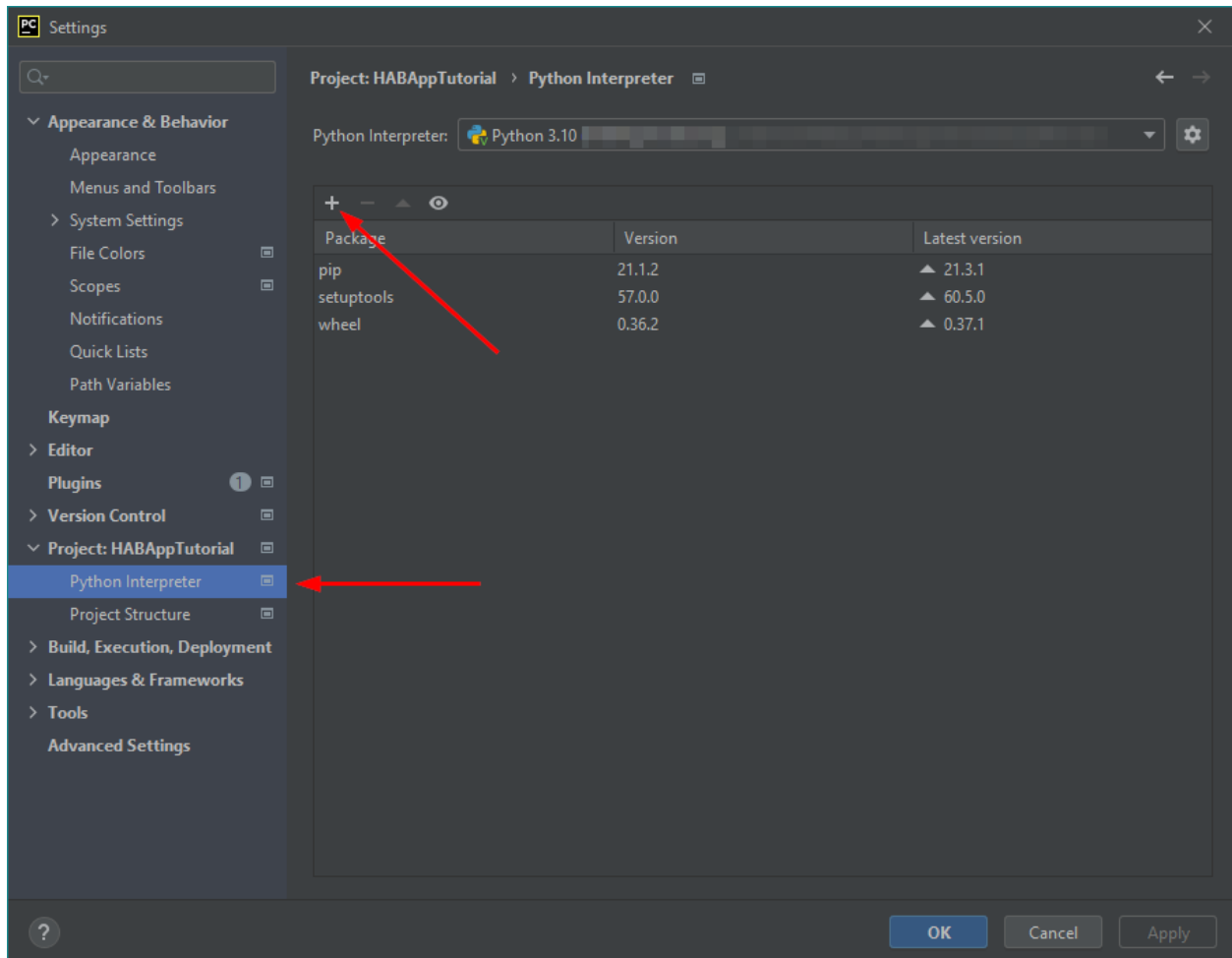
1.5.1 Type hints and checks

To enable type hints and checks HABApp needs to be installed in the python environment that is currently used by PyCharm. Ensure that the HABApp version for PyCharm matches the HABApp version that is currently deployed and running the rules. It is recommended to create a new virtual environment when creating a new project for HABApp.

Go to Settings and view the current python environment settings.

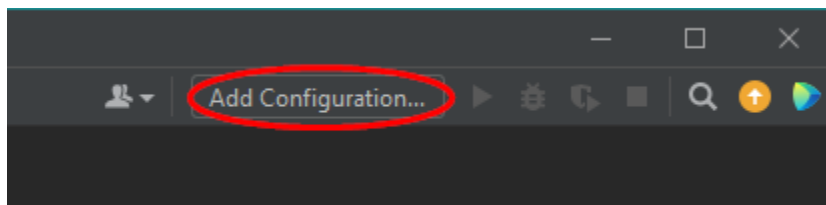


Install the HABApp package through the + symbol. Once the installation was successful PyCharm will provide checks and hints.

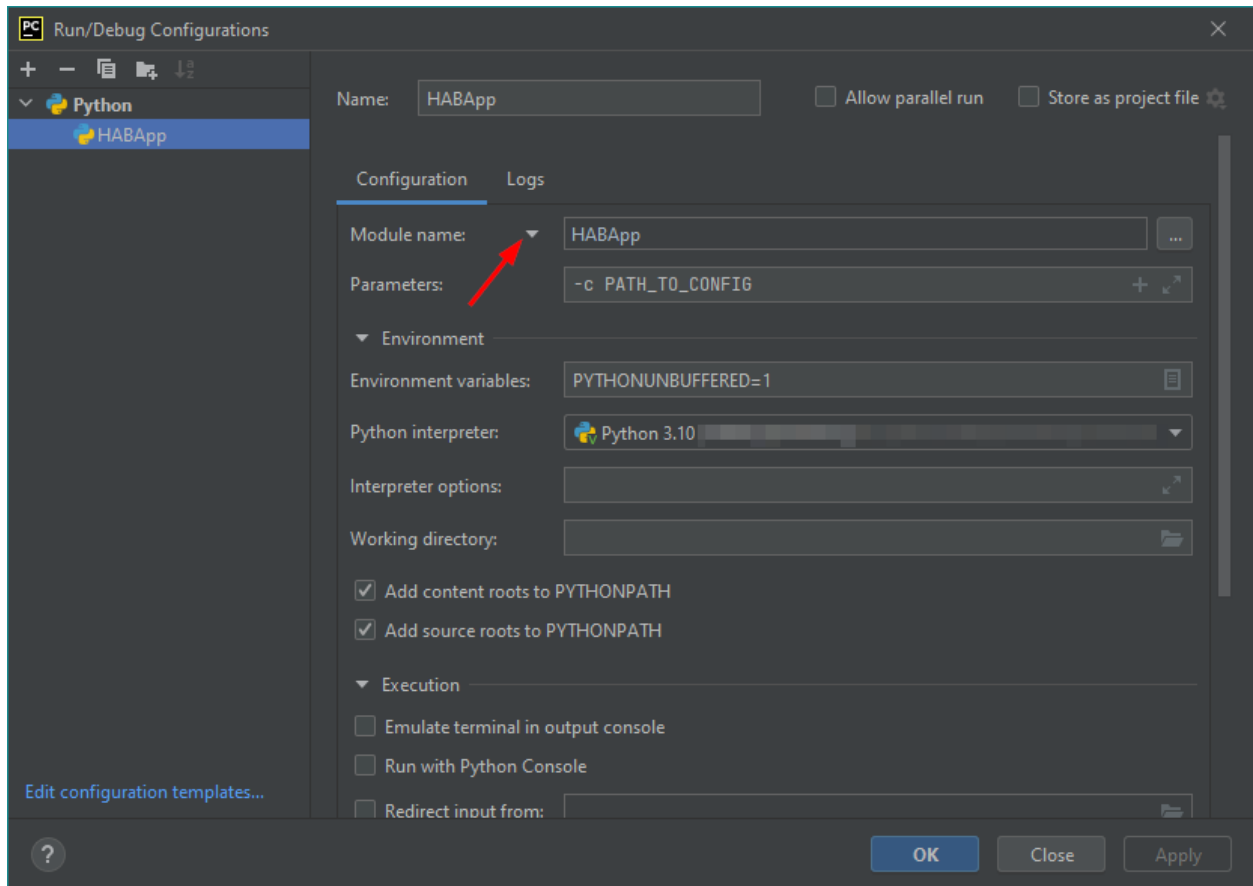


1.5.2 Start HABApp from PyCharm

It is possible to start HABApp directly from pycharm e.g. to debug things. Open the run configurations.



Switch to **Module** name execution with the small dropdown arrow. It's still necessary to supply a configuration file which can be done in the **Parameters** line.



After a click on “OK” HABApp can be run/debugged directly from pycharm.
It’s even possible to create breakpoints in rules and inspect all objects.

1.6 Install a development version of HABApp

To try out new features or test some functionality it’s possible to install a branch directly from github. Installation works only in a virtual environment.

New features are typically first available in the `DeveloP` branch.

1. Navigate to the folder where the virtual environment was created:

```
cd /opt/habapp
```

2. Activate the virtual environment

Linux:

```
source bin/activate
```

Windows:

```
Scripts\activate
```

3. Remove existing HABApp installation:

```
python3 -m pip uninstall habapp
```

4. Install HABApp from the github branch (here Develop):

```
python3 -m pip install git+https://github.com/spacemanspiff2007/HABApp.git@Develop
```

5. Run HABApp as usual (e.g. through systemctl) or manually with:

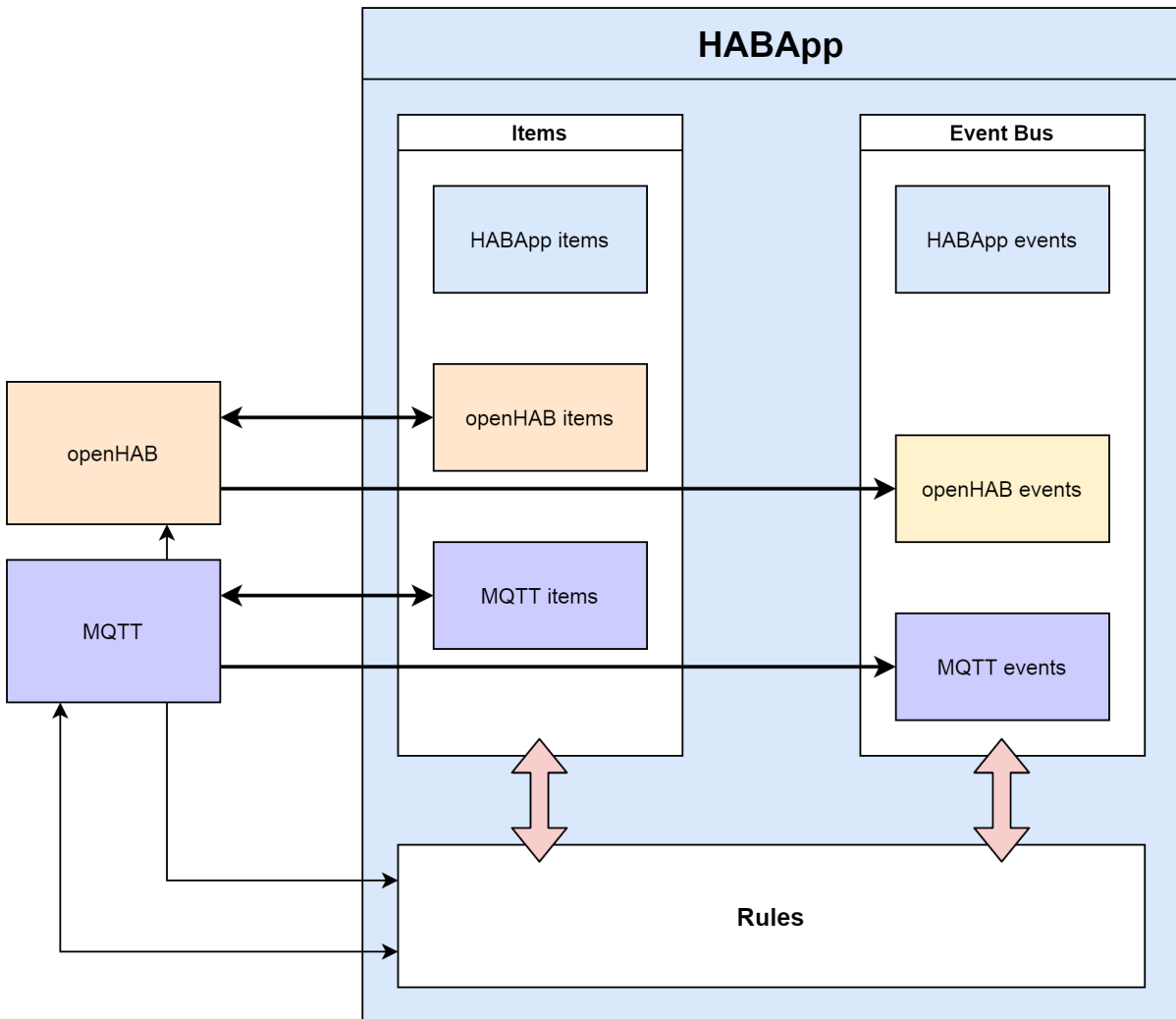
```
habapp --config PATH_TO_CONFIGURATION_FOLDER
```


ABOUT HABAPP

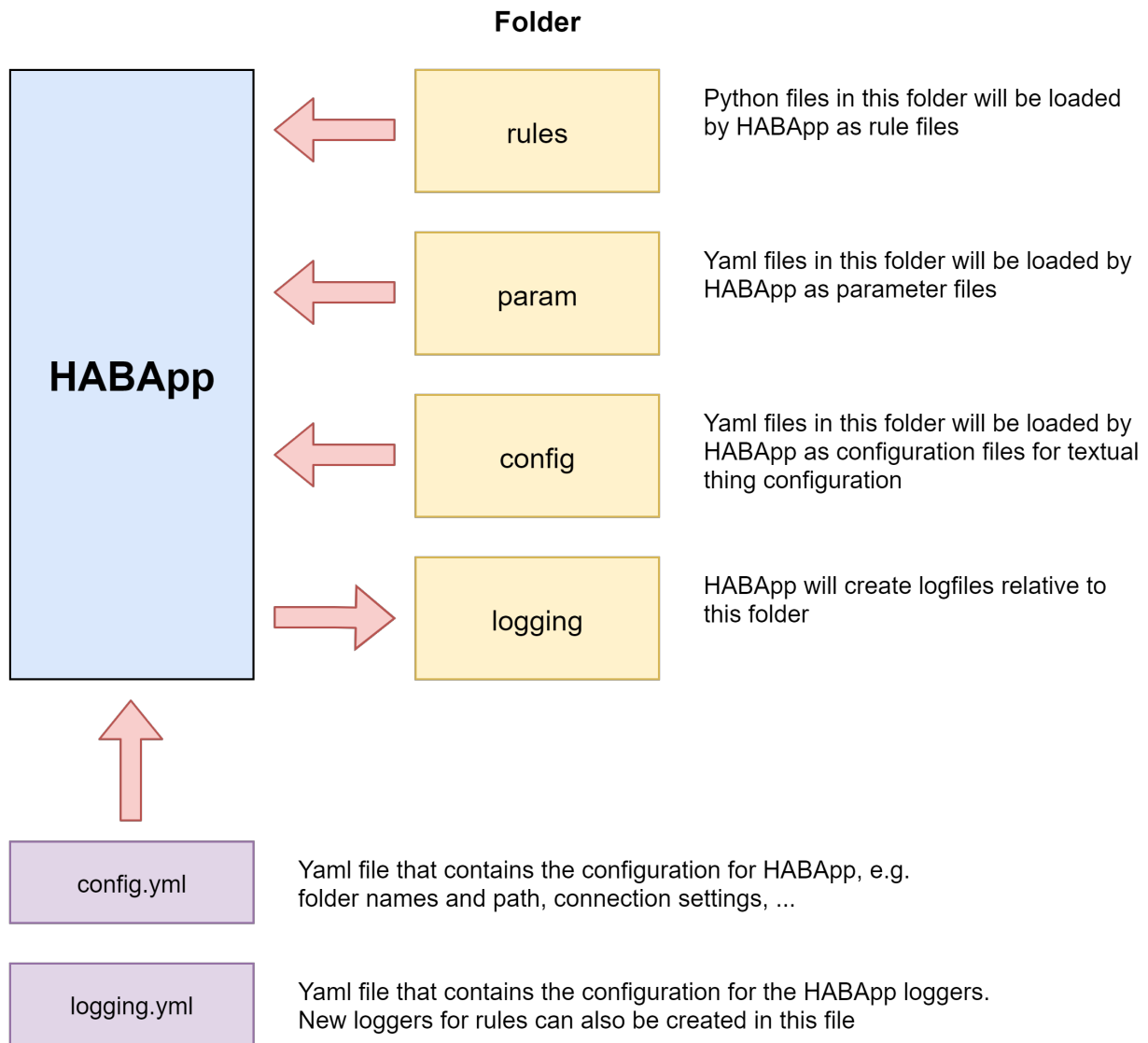
2.1 About

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.

CONFIGURATION

3.1 Description

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

3.2 Example

```
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
```

(continues on next page)

(continued from previous page)

```

interval: 10          # Seconds between two pings

connection:
  url: http://localhost: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:
    identifier: HABApp
    host: ''
    port: 8883
    user: ''
    password: ''
    tls:
      enabled: false  # Enable TLS for the connection
      insecure: false # Validate server hostname in server certificate
      ca cert: ''     # Path to a CA certificate that will be treated as trusted
                      # (e.g. when using a self signed certificate)

  subscribe:          # Changes to Subscribe get picked up without restarting HABApp
    qos: 0             # Default QoS for subscribing
    topics:
      - '#'            # Subscribe to this topic, qos is default QoS
      - ['my/topic', 1] # Subscribe to this topic with explicit QoS

  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.

```

It's possible to use environment variables and files (e.g. docker secrets) in the configuration. See [the easyconfig documentation](#) for the exact syntax and examples.

3.3 Configuration Reference

All possible configuration options are described here. Not all entries are created by default in the config file and one should take extra care when changing those entries.

settings ApplicationConfig

Structure that contains the complete configuration

field directories: *DirectoriesConfig* [Optional]

field habapp: *HABAppConfig* [Optional]

field location: *LocationConfig* [Optional]

field mqtt: *MqttConfig* [Optional]

field openhab: *OpenhabConfig* [Optional]

3.3.1 Directories

settings DirectoriesConfig

Configuration of directories that are used

field config: *Optional[Path]* = 'config'

Folder from which configuration files (e.g. for textual thing configuration) will be loaded

field lib: *Optional[Path]* = 'lib'

Folder where additional libraries can be placed

field logging: *Path* = 'log'

Folder where the logs will be written to

field param: *Optional[Path]* = 'params'

Folder from which the parameter files will be loaded

field rules: *Path* = 'rules'

Folder from which the rule files will be loaded

classmethod ensure_folder(*value*)

3.3.2 Location

settings LocationConfig

location where the instance is running. Is used to calculate Sunrise/Sunset.

field elevation: *float* = 0.0

field latitude: *float* = 0.0

field longitude: *float* = 0.0

3.3.3 MQTT

settings MqttConfig

MQTT configuration

field connection: [Connection](#) [Optional]

field general: [General](#) [Optional]

field publish: [Publish](#) [Optional]

field subscribe: [Subscribe](#) [Optional]

Connection

settings Connection

field host: `str = ''`

Connect to this host. Empty string ("") disables the connection.

field identifier: `str = 'HABApp-FywVPmodD0zdA'`

Identifier that is used to uniquely identify this client on the mqtt broker.

field password: `str = ''`

field port: `int = 1883`

field tls: [TLSSettings](#) [Optional]

field user: `str = ''`

TLS

settings TLSSettings

field ca cert: `Path = ''`

Path to a CA certificate that will be treated as trusted

field enabled: `bool = True`

Enable TLS for the connection

field insecure: `bool = False`

Validate server hostname in server certificate

Subscribe

settings Subscribe

field qos: `Literal[0, 1, 2] = 0`

Default QoS for subscribing

field topics: `Tuple[Tuple[str, Optional[Literal[0, 1, 2]]], ...] = ('#',)`

Publish

settings Publish

field qos: `Literal[0, 1, 2] = 0`

Default QoS when publishing values

field retain: `bool = False`

Default retain flag when publishing values

General

settings General

field listen_only: `bool = False`

If True HABApp does not publish any value to the broker

3.3.4 Openhab

settings OpenhabConfig

field connection: `Connection [Optional]`

field general: `General [Optional]`

field ping: `Ping [Optional]`

Connection

settings Connection

field buffer: `ByteSize = '128kib'`

Buffer for reading lines in the SSE event handler. This is the buffer that gets allocated for every(!) request and SSE message that the client processes. Increase only if you get error messages or disconnects e.g. if you use large images.

field password: `str = ''`

field topic filter: `str = 'openhab/items/*,openhab/channels/*,openhab/things/*'`

Topic filter for subscribing to openHAB. This filter is processed by openHAB and only events matching this filter will be sent to HABApp.

field url: `str = 'http://localhost:8080'`

Connect to this url. Empty string ("") disables the connection.

field user: `str = ''`

field verify_ssl: `bool = True`

Check certificates when using https

Ping

settings Ping

field enabled: `bool` = `True`

If enabled the configured item will show how long it takes to send an update from HABApp and get the updated value back from openHAB in milliseconds

field interval: `Union[int, float]` = `10`

Seconds between two pings

Constraints

- `ge` = 0.1

field item: `str` = `'HABApp_Ping'`

Name of the Numberitem

General

settings General

field listen_only: `bool` = `False`

If True HABApp does not change anything on the openHAB instance.

field min_start_level: `int` = `70`

Minimum openHAB start level to load items and listen to events

Constraints

- `ge` = 0
- `le` = 100

field min_uptime: `int` = `60`

Minimum openHAB uptime in seconds to load items and listen to events

Constraints

- `ge` = 0
- `le` = 3600

field wait_for_openhab: `bool` = `True`

If True HABApp will wait for a successful openHAB connection before loading any rules on startup

3.3.5 HABApp

settings HABAppConfig

HABApp internal configuration. Only change values if you know what you are doing!

field logging: `LoggingConfig` [Optional]

field thread pool: `ThreadPoolConfig` [Optional]

ThreadPool

settings ThreadPoolConfig

field enabled: `bool` = `True`

When the thread pool is disabled HABApp will become an asyncio application. Use only if you have experience developing asyncio applications! If the thread pool is disabled using blocking calls in functions can and will break HABApp

field threads: `Annotated[int]` = `10`

Amount of threads to use for the executor

Constraints

- `ge` = 1
- `le` = 16

Logging

settings LoggingConfig

field flush every: `float` = `0.5`

Wait time in seconds before the buffer gets flushed again when it was empty

Constraints

- `ge` = 0.1

field use buffer: `bool` = `True`

Automatically inject a buffer for the event log

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 development.

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.

4.3.1 Access

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")
```

4.3.2 Values

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

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()
```

```
[ HABApp.Items]    DEBUG | Added Item_Name (Item)
[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"
```

4.3.3 Timestamps

All items have two additional timestamps set which can be used to simplify rule logic.

- The time when the item was last updated
- The time when the item was last changed.

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

class TimestampRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        # This item was created by another rule, that's why "get_item" is used
        self.my_item = Item.get_item('Item_Name')

        # Access of timestamps
        print(f'Last update: {self.my_item.last_update}')
        print(f'Last change: {self.my_item.last_change}')

TimestampRule()
```

```
Last update: 2022-08-20T12:16:00
Last change: 2022-08-20T10:30:00
```

4.4 Watch items for events

It is possible to watch items for changes or updates. The `listen_event` function takes an instance of `EventFilter` which describes the kind of event that will be passed to the callback.

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

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')

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

        # If you already have an item you can use the more convenient method of the item
        # This is the recommended way to use the event listener
        self.my_item.listen_event(self.item_updated, ValueUpdateEventFilter())

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

(continues on next page)

(continued from previous page)

```
# the function has 1 argument which is the event
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}')

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}')
```

```
MyFirstRule()
```

```
Item_Name updated value: "Changed value"
Last update of Item_Name: 2024-04-12T12:54:52.613457
Item_Name updated value: "Changed value"
Last update of Item_Name: 2024-04-12T12:54:52.613457
Item_Name changed from "Some value" to "Changed value"
Last change of Item_Name: 2024-04-12T12:54:52.613457
```

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)

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

```
MyFirstRule()
```

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

4.6 Convenience functions

HABApp provides some convenience functions which make the rule creation easier and reduce boiler plate code.

4.6.1 post_value_if

`post_value_if` will post a value to the item depending on its current state. There are various comparisons available (see [documentation](#)) Something similar is available for openHAB items (`oh_post_update_if`)

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

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):

        # This construct
        if self.my_item != 'overwrite value':
            self.my_item.post_value('Test')

        # ... is equivalent to
        self.my_item.post_value_if('Test', not_equal='overwrite value')

        # This construct
        if self.my_item == 'overwrite value':
            self.my_item.post_value('Test')

        # ... is equivalent to
        self.my_item.post_value_if('Test', equal='overwrite value')

MyFirstRule()
```

LOGGING

5.1 Configuration

Configuration of logging is done through the `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 an absolute path as a file name, at least for the `HABApp.log`. That way even if the HABApp configuration is invalid HABApp can still log the errors that have occurred.
e.g.: `/HABApp/logs/habapp.log` or `c:\HABApp\logs\habapp.log`

5.2 Provided loggers

The `HABApp.config.logging` module provides additional loggers which can be used

class `MidnightRotatingFileHandler(*args, **kwargs)`

A rotating file handler that checks once after midnight if the configured size has been exceeded and then rotates the file

class `CompressedMidnightRotatingFileHandler(*args, **kwargs)`

Same as `MidnightRotatingFileHandler` but rotates the file to a gzipped archive (`.gz`)

5.3 Example

5.3.1 Usage

The logging library is the standard python library and an extensive description can be found [in the official documentation](#).

```
import logging
import HABApp
```

(continues on next page)

(continued from previous page)

```
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()
```

To make the logging output work properly an output file and an output format has to be configured for the logger. The logging library supports a logging hierarchy so the configuration for the logger `MyRule` will also work logger `MyRule.SubLogger` and `MyRule.SubLogger.SubSubLogger`.

The output of our logger from the example shall be in a separate file so we add a new output file to the file configuration under handlers in the `logging.yml`.

```
handlers:
    ...

    MyRuleHandler: # <-- This is the name of the handler
        class: HABApp.config.logging.MidnightRotatingFileHandler
        filename: 'c:\HABApp\Logs\MyRule.log'
        maxBytes: 10_000_000
        backupCount: 3

        formatter: HABApp_format # use this format
        level: DEBUG
```

The output file is now available for logging but the configuration for the logger is still missing. It has to be added under loggers and reference the handler we created

```
loggers:
    ...

    MyRule:
        # <-- Name of the logger
        level: DEBUG # <-- minimum Logging level, e.g. use INFO if you don't want the
        ↪ output of log.debug()
        handlers:
            - MyRuleHandler # This logger uses the MyRuleHandler
        propagate: False
```

Now the logger works as expected and writes all output to the new file.

5.3.2 Full Example configuration

```
# -----
# Configuration of the available output formats
# -----
formatters:
  HABApp_format:
    format: '%(asctime)s] [%(name)25s] %(levelname)8s | %(message)s'

# -----
# Configuration of the available file handlers (output files)
# -----
handlers:
  HABApp_default:
    class: HABApp.config.logging.MidnightRotatingFileHandler
    filename: 'HABApp.log'
    maxBytes: 10_000_000
    backupCount: 3

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

  MyRuleHandler:
    class: HABApp.config.logging.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

# -----
# Configuration of all available loggers and their configuration
# -----
loggers:
  HABApp:
    level: DEBUG
    handlers:
      - HABApp_default # This logger does log with the default handler
    propagate: False

  MyRule: # <-- Name of the logger
    level: DEBUG
    handlers:
      - MyRuleHandler # This logger uses the MyRuleHandler
    propagate: False
```

5.4 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.

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

formatters:
  HABApp_format:
  ...
```

5.5 Logging to stdout

The following handler writes to stdout

```
handlers:
  StdOutHandler:
    class: logging.StreamHandler
    stream: ext://sys.stdout

    formatter: HABApp_format
    level: DEBUG
```

5.6 Add custom filters to loggers

It's possible to filter out certain parts of log files with a `filter`. The recommendation is to create the filter *during startup*.

This example ignores all messages for the `HABApp.EventBus` logger that contain `MyIgnoredString`.

```
import logging

# False to skip, True to log record
def filter(record: logging.LogRecord) -> bool:
    return 'MyIgnoredString' not in record.msg

logging.getLogger('HABApp.EventBus').addFilter(filter)
```

Note:

Regular expressions for a filter should be compiled outside of the filter function with `re.compile` for performance reasons.

A simple subtext search however will always have way better performance.

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 Interacting with 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 be passed as an argument into the function.

There is the possibility to reduce the function calls to a certain event type with an additional event filter (typically `ValueUpdateEventFilter` or `ValueChangeEventFilter`).

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, \
    ValueChangeEventFilter, ValueUpdateEventFilter
from HABApp.core.items import Item

class MyRule(Rule):
    def __init__(self):
        super().__init__()
        self.listen_event('MyOpenhabItem', self.on_change, ValueChangeEventFilter()) #_
        # trigger only on ValueChangeEvent
        self.listen_event('My/MQTT/Topic', self.on_update, ValueUpdateEventFilter()) #_
        # 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, ValueUpdateEventFilter())

    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 the value to the event filter. There are convenience Filters (e.g. `ValueUpdateEventFilter` and `ValueChangeEventFilter`) for the most used event types that provide type hints.

6.2.1 NoEventFilter

class NoEventFilter
Triggers on all events

6.2.2 EventFilter

class EventFilter(*event_class*, ***kwargs*)
Triggers on event types and optionally on their values, too

6.2.3 ValueUpdateEventFilter

class ValueUpdateEventFilter(*value=<MISSING>*)

6.2.4 ValueChangeEventFilter

class ValueChangeEventFilter(*value=<MISSING>*, *old_value=<MISSING>*)

6.2.5 AndFilterGroup

class AndFilterGroup(**args*)
All child filters have to match

6.2.6 OrFilterGroup

class OrFilterGroup(**args*)
Only one child filter has to match

6.2.7 Example

Example

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

class MyRule(Rule):
    def __init__(self):
        super().__init__()
        my_item = Item.get_item('MyItem')

        # This will only call the callback for ValueUpdateEvents
        my_item.listen_event(self.on_val_my_value, ValueUpdateEventFilter())

        # This will only call the callback for ValueUpdateEvents where the value==my_
```

(continues on next page)

(continued from previous page)

```

↪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'))

    # trigger if the value is 1 or 2 by using both filters with or
    my_item.listen_event(
        self.value_1_or_2,
        OrFilterGroup(
            ValueUpdateEventFilter(value=1), ValueUpdateEventFilter(value=2)
        )
    )

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

    def value_1_or_2(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
<code>soon()</code>	Run the callback as soon as possible (typically in the next second).
<code>at()</code>	Run the callback in x seconds or at a specified time.
<code>countdown()</code>	Run a function after a time has run down
<code>every()</code>	Run a function periodically
<code>every_minute()</code>	Run a function every minute
<code>every_hour()</code>	Run a function every hour
<code>on_every_day()</code>	Run a function at a specific time every day
<code>on_workdays()</code>	Run a function at a specific time on workdays
<code>on_weekends()</code>	Run a function at a specific time on weekends
<code>on_day_of_week()</code>	Run a function at a specific time on specific days of the week
<code>on_sun_dawn()</code>	Run a function on dawn
<code>on_sunrise()</code>	Run a function on sunrise
<code>on_sunset()</code>	Run a function on sunset
<code>on_sun_dusk()</code>	Run a function on dusk

All functions return an instance of `ScheduledCallbackBase`

```
class HABAppSchedulerView(context)
```

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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – 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** (Callable[[ParamSpec(HINT_CB_P)], Any]) – Function which will be called
- **args** (ParamSpecArgs) – Positional arguments that will be passed to the function
- **kwargs** (ParamSpecKwargs) – 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** (Callable[[ParamSpec(HINT_CB_P)], Any]) – Function which will be called
- **args** (ParamSpecArgs) – Positional arguments that will be passed to the function
- **kwargs** (ParamSpecKwargs) – 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** (Callable[[ParamSpec(HINT_CB_P)], Any]) – Function which will be called
- **args** (ParamSpecArgs) – Positional arguments that will be passed to the function
- **kwargs** (ParamSpecKwargs) – 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** (Callable[[ParamSpec(HINT_CB_P)], Any]) – Function which will be called
- **args** (ParamSpecArgs) – Positional arguments that will be passed to the function
- **kwargs** (ParamSpecKwargs) – 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`**every_hour**(*callback, *args, **kwargs*)

Picks a random minute and second and run the callback every hour

Parameters

- **callback** (`Callable[[ParamSpec(HINT_CB_P)], Any]`) – Function which will be called
- **args** (`ParamSpecArgs`) – Positional arguments that will be passed to the function
- **kwargs** (`ParamSpecKwargs`) – Keyword arguments that will be passed to the function

Return type`ReoccurringJob`

6.4 Other tools and scripts

HABApp provides convenience functions to run other tools and scripts. The working directory for the new process is by default the folder of the HABApp configuration file.

6.4.1 Running tools

External tools can be run with the `execute_subprocess()` function. Once the process has finished the callback will be called with the captured output of the process. Example:

```
import HABApp

class MyExecutionRule(HABApp.Rule):

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

        self.execute_subprocess( self.func_when_finished, 'path_to_program', 'arg1_for_
↪program')

    def func_when_finished(self, process_output: str):
        print(process_output)

MyExecutionRule()
```

6.4.2 Running python scripts or modules

Python scripts can be run with the `execute_python()` function. The working directory for a script is by default the folder of the script. Once the script or module has finished the callback will be called with the captured output of the module/script. Example:

```
import HABApp

class MyExecutionRule(HABApp.Rule):

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

        self.execute_python( self.func_when_finished, '/path/to/python/script.py', 'arg1_
↪for_script')

    def func_when_finished(self, module_output: str):
        print(module_output)

MyExecutionRule()
```

6.4.3 FinishedProcessInfo

It's possible to get the raw process output instead of just the captured string. See `execute_subprocess()` or `execute_python()` on how to enable it.

class FinishedProcessInfo(*returncode, stdout, stderr*)

Information about the finished process.

Variables

- **returncode** (*int*) – Return code of the process

- `stdout` (*Optional[str]*) – Standard output of the process or None
- `stderr` (*Optional[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

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 Rule

Variables

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

on_rule_loaded()

Override this to implement logic that will be called when the rule and the file has been successfully loaded

on_rule_removed()

Override this to implement logic that will be called when the rule has been unloaded.

post_event(name, event)

Post an event to the event bus

Parameters

- **name** (`Union[TypeVar(HINT_ITEM_OBJ, bound= BaseItem), str]`) – name or item to post event to
- **event** (`Any`) – Event class to be used (must be class instance)

Returns

listen_event(name, callback, event_filter=None)

Register an event listener

Parameters

- **name** (`Union[TypeVar(HINT_ITEM_OBJ, bound= BaseItem), str]`) – item or name to listen to
- **callback** (`Callable[[Any], Any]`) – callback that accepts one parameter which will contain the event
- **event_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound= EventFilterBase)]`) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)`

execute_subprocess(callback, program, *args, additional_python_path=None, capture_output=True, raw_info=False, **kwargs)

Run another program

Parameters

- **callback** – Function that will be called when the process has finished. First parameter takes a `str` when `raw_info` is `False` (default) else an instance of *FinishedProcessInfo*
- **program** (`Union[str, Path]`) – python module (path to file) or python package

- **args** (`Union[str, Path]`) – arguments passed to the module or to package
- **raw_info** (`bool`) – `False`: Return only the textual process output. In case of failure (return code `!= 0`) a log entry and an error event will be created. This is the default and should be fine for almost all use cases.

`True`: The callback will always be called with an instance of `FinishedProcessInfo`.
- **capture_output** (`bool`) – Capture program output, set to `False` to only capture the return code
- **additional_python_path** (`Optional[Iterable[Union[str, Path]]]`) – additional folders which will be added to the env variable `PYTHONPATH`
- **kwargs** – Additional kwargs that will be passed to `asyncio.create_subprocess_exec`

Returns

execute_python(*callback, module_or_package, *args, additional_python_path=None, capture_output=True, raw_info=False, **kwargs*)

Run a python module or package as a new process. The python environment that is used to run HABApp will be to run the module or package.

Parameters

- **callback** – Function that will be called when the process has finished. First parameter takes a `str` when `raw_info` is `False` (default) else an instance of `FinishedProcessInfo`
- **module_or_package** (`Union[str, Path]`) – python module (path to file) or python package (just the name)
- **args** (`Union[str, Path]`) – arguments passed to the module or to package
- **raw_info** (`bool`) – `False`: Return only the textual process output. In case of failure (return code `!= 0`) a log entry and an error event will be created. This is the default and should be fine for almost all use cases.

`True`: The callback will always be called with an instance of `FinishedProcessInfo`.
- **capture_output** (`bool`) – Capture program output, set to `False` to only capture the return code
- **additional_python_path** (`Optional[Iterable[Union[str, Path]]]`) – additional folders which will be added to the env variable `PYTHONPATH`
- **kwargs** – Additional kwargs that will be passed to `asyncio.create_subprocess_exec`

Returns

static get_items(*type=None, name=None, tags=None, groups=None, metadata=None, metadata_value=None*)

Search the HABApp item registry and return the found items.

Parameters

- **type** (`Union[Tuple[Type[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)], ...], Type[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)], None]`) – item has to be an instance of this class
- **name** (`Union[str, Pattern[str], None]`) – str (will be compiled) or regex that is used to search the Name
- **tags** (`Union[str, Iterable[str], None]`) – item must have these tags (will return only instances of `OpenhabItem`)

- **groups** (`Union[str, Iterable[str], None]`) – item must be a member of these groups (will return only instances of `OpenhabItem`)
- **metadata** (`Union[str, Pattern[str], None]`) – str (will be compiled) or regex that is used to search the metadata (e.g. ‘homekit’)
- **metadata_value** (`Union[str, Pattern[str], None]`) – str (will be compiled) or regex that is used to search the metadata value (e.g. ‘TargetTemperature’)

Return type

`Union[List[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)], List[BaseItem]]`

Returns

Items that match all the passed criteria

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 `Parameter` and `DictParameter` available.

```
from HABApp import Rule, Parameter
from HABApp.core.events import ValueChangeEventFilter

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

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

        # deeper structuring is possible through specifying multiple keys
        self.min_value_nested = 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, ValueChangeEventFilter())

    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.

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

Parameters 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

```
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 `Parameter(filename, *keys, default_value='ToDo')`

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.

class `DictParameter(filename, *keys, default_value='ToDo')`

Implements a dict interface

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.

This page describes the HABApp internals

8.1 Datatypes

HABApp provides some datatypes that simplify e.g. the color handling.

8.1.1 RGB

Datatype for RGB (red, green, blue) color handling. RGB types can be sent directly to openHAB and will be converted accordingly. Additionally there are wider RGB types (e.g. RGB16, RGB32) available.

```
from HABApp.core.types import RGB

col = RGB(5, 15, 255)
print(col)

print(col.red)    # red value
print(col.r)      # short name for red value
print(col[0])     # access of red value through numeric index

new_col = col.replace(red=22)
print(new_col)
print(new_col.to_hsb())
```

```
RGB(5, 15, 255)
5
5
5
RGB(22, 15, 255)
HSB(241.75, 94.12, 100.00)
```

```
class RGB(r, g, b)
```

```
    classmethod from_hsb(obj)
```

Return new Object from a HSB object for a hsb tuple

Parameters

obj (`Union[HSB, Tuple[float, float, float]]`) – HSB object or tuple with HSB values

Return type

Self

Returns

new RGB object

replace(*r=None, g=None, b=None, red=None, green=None, blue=None*)

Create a new object with (optionally) replaced values.

Parameters

- **r** (*Optional[int]*) – new red value
- **red** (*Optional[int]*) – new red value
- **g** (*Optional[int]*) – new green value
- **green** (*Optional[int]*) – new green value
- **b** (*Optional[int]*) – new blue value
- **blue** (*Optional[int]*) – new blue value

Return type

Self

to_hsb()

Create a new HSB object from this object

Return type*HSB***Returns**

New HSB object

property b: *int*

blue value

property blue: *int*

blue value

property g: *int*

green value

property green: *int*

green value

property r: *int*

red value

property red: *int*

red value

8.1.2 HSB

Datatype for HSB (hue, saturation, brightness) color handling. HSB types can be sent directly to openHAB and will be converted accordingly.

```
from HABApp.core.types import HSB

col = HSB(200, 25, 75)
print(col)

print(col.hue)    # hue value
print(col.h)      # short name for hue value
print(col[0])     # access of hue value through numeric index

new_col = col.replace(hue=22)
print(new_col)
print(new_col.to_rgb())
```

```
HSB(200.00, 25.00, 75.00)
200
200
200
HSB(22.00, 25.00, 75.00)
RGB(191, 161, 143)
```

class `HSB(hue, saturation, brightness)`

classmethod `from_rgb(obj)`

Create an HSB object from an RGB object or an RGB tuple

Parameters

obj (`Union[RGB, Tuple[int, int, int]]`) – HSB object or RGB tuple

Return type

`Self`

Returns

New HSB object

replace(*h=None, s=None, b=None, hue=None, saturation=None, brightness=None*)

Create a new object with (optionally) replaced values.

Parameters

- **h** (`Optional[float]`) – New hue value
- **hue** (`Optional[float]`) – New hue value
- **s** (`Optional[float]`) – New saturation value
- **saturation** (`Optional[float]`) – New saturation value
- **b** (`Optional[float]`) – New brightness value
- **brightness** (`Optional[float]`) – New brightness value

Return type

`Self`

to_rgb()

Create an RGB object from this object

Return type

RGB

Returns

New RGB object

property b: *float*

brightness value

property brightness: *float*

brightness value

property h: *float*

hue value

property hue: *float*

hue value

property s: *float*

saturation value

property saturation: *float*

saturation value

8.2 Items

8.2.1 Item



class Item()

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

Return type

Item

Returns

The 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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

`Any`

Returns

value of the item

listen_event(`callback, event_filter=None`)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=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

post_value_if(`new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)`

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post

- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

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

property name: `str`

Returns

Name of the item (read only)

8.2.2 ColorItem



class ColorItem()

Item for dealing with color related values

classmethod get_create_item(*name*, *hue=0.0*, *saturation=0.0*, *brightness=0.0*)

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_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. This is a helper function that returns a default in case the item value is None.

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_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=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 %)

post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type*bool***Returns***True* if the new value was posted else *False*

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: DateTime

Returns

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

property last_update: DateTime

Returns

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

property name: str

Returns

Name of the item (read only)

8.2.3 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.

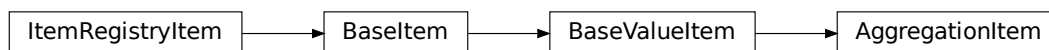
```
from HABApp.core.items import AggregationItem
my_agg = AggregationItem.get_create_item('MyAggregationItem')

# 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 AggregationItem()

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 and 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 item 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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* which will also trigger on changes/update from open-hab or mqtt. Additionally it can be an instance of *EventFilter* which additionally filters on the values of the event. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type`TypeVar(HINT_EVENT_BUS_LISTENER, bound= 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

post_value_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. `None`)
- **is_not** – item state has to be not the same object as the passt value (e.g. `None`)

Return type`bool`**Returns**

True if the new value was posted else *False*

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: DateTime**Returns**

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

property last_update: DateTime**Returns**

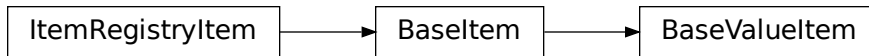
Timestamp of the last time when the item has been updated (read only)

property name: str**Returns**

Name of the item (read only)

8.2.4 BaseValueItem

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



class BaseValueItem()

Simple item

Variables

- **name** (*str*) – Name of the item (read only)
- **value** – Value of the item, can be anything (read only)
- **last_change** (*datetime*) – Timestamp of the last time when the item has changed the value (read only)
- **last_update** (*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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(callback, event_filter=None)

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_filter** (*Optional*[*TypeVar*(*HINT_EVENT_FILTER_OBJ*, bound=*EventFilterBase*)]]) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* which will also trigger on changes/update from open-hab or mqtt. Additionally it can be an instance of *EventFilter* which additionally filters on the values of the event. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(*HINT_EVENT_BUS_LISTENER*, bound=*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

post_value_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is_*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else False

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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

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

property name: `str`

Returns

Name of the item (read only)

8.3 Events

8.3.1 ValueUpdateEvent

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

ValueUpdateEvent

```
class ValueUpdateEvent(name, value)
```

Variables

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

8.3.2 ValueChangeEvent

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

ValueChangeEvent

```
class ValueChangeEvent(name, value, old_value)
```

Variables

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

8.3.3 ItemNoUpdateEvent

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

ItemNoUpdateEvent

```
class ItemNoUpdateEvent(name, seconds)
```

Variables

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

8.3.4 ItemNoChangeEvent

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

ItemNoChangeEvent

```
class ItemNoChangeEvent(name, seconds)
```

Variables

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

9.1 Additional configuration

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.

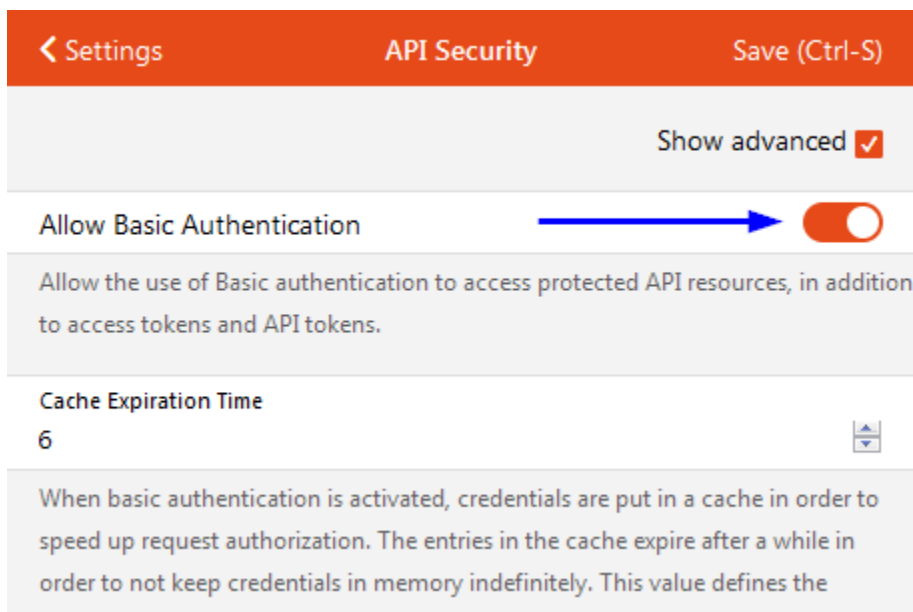
9.1.1 Textual configuration

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

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

9.1.2 GUI

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



9.2 openHAB item types

9.2.1 Description and example

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

Example:

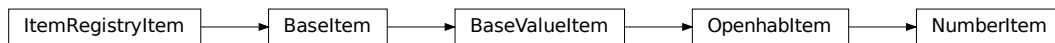
```
from HABApp.openhab.items import ContactItem, SwitchItem

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!
```

9.2.2 NumberItem



```
class NumberItem()
```

NumberItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*Union[int, float]*) – Current item value (or state in openHAB wording)
- **label** (*Optional[str]*) – Item label or None if not configured
- **tags** (*FrozenSet[str]*) – Item tags
- **groups** (*FrozenSet[str]*) – The groups the item is in
- **metadata** (*Mapping[str, MetaData]*) – Item metadata

```
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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (*Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]*) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value

- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(*value*=<MISSING>)

Send a command to the openHAB item

Parameters**value** (*Any*) – (optional) value to be sent. If not specified the current 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**post_value_if**(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value

- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: DateTime**Returns**

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

property last_update: DateTime**Returns**

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

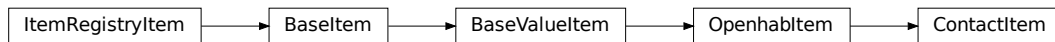
property name: str**Returns**

Name of the item (read only)

property unit: str | None

Return the item unit if it is a “Unit of Measurement” item else None

9.2.3 ContactItem

**class ContactItem()****Variables**

- **name** (*str*) – Item name
- **value** (*str*) – Current item value (or state in openHAB wording)
- **label** (*Optional[str]*) – Item label or None if not configured
- **tags** (*FrozenSet[str]*) – Item tags
- **groups** (*FrozenSet[str]*) – The groups the item is in
- **metadata** (*Mapping[str, Metadata]*) – Item metadata

closed()

Post an update to the item with the closed value

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. This is a helper function that returns a default in case the item value is None.

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_filter=None*)

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_filter** (*Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]*) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

```
oh_post_update_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
                  ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>,
                  lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>,
                  gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>,
                  is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

```
oh_send_command(value=<MISSING>)
```

Send a command to the openHAB item

Parameters

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

```
open()
```

Post an update to the item with the open value

```
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

```
post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
               ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>,
               le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>,
               greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else False

```
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: DateTime**Returns**

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

property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.4 SwitchItem

**class SwitchItem()**

`SwitchItem` which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name

- **value** (*str*) – Current item value (or state in openHAB wording)
- **label** (*Optional[str]*) – Item label or None if not configured
- **tags** (*FrozenSet[str]*) – Item tags
- **groups** (*FrozenSet[str]*) – The groups the item is in
- **metadata** (*Mapping[str, MetaData]*) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

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_filter=None*)

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_filter** (*Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]*) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. [AndFilterGroup](#) and [OrFilterGroup](#)

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)`

off()

Command item off

oh_post_update(value=<MISSING>)

Post an update to the openHAB item

Parameters

value ([Any](#)) – (optional) value to be posted. If not specified the current item value will be used.

oh_post_update_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

oh_send_command(*value*=<MISSING>)

Send a command to the openHAB item

Parameters

value (*Any*) – (optional) value to be sent. If not specified the current 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

post_value_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

bool

Returns

True if the new value was posted else *False*

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

toggle()

Toggle the switch. Turns the switch on when off or off when currently on.

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: DateTime**Returns**

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

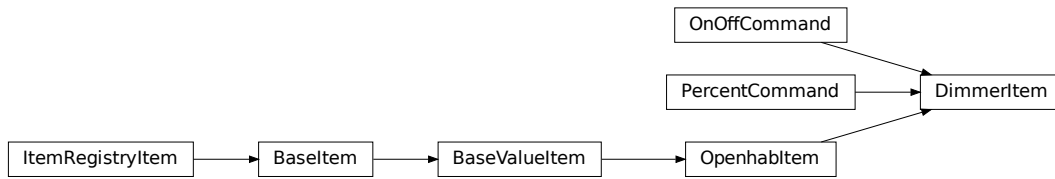
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.5 DimmerItem



class `DimmerItem()`

DimmerItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*Union[int, float]*) – Current item value (or state in openHAB wording)
- **label** (*Optional[str]*) – Item label or None if not configured
- **tags** (*FrozenSet[str]*) – Item tags
- **groups** (*FrozenSet[str]*) – The groups the item is in
- **metadata** (*Mapping[str, MetaData]*) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

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_filter*=None)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)`

off()

Command item off

oh_post_update(*value*=<MISSING>)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is_*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value

- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(*value*=<MISSING>)

Send a command to the openHAB item

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

post_value_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value

- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

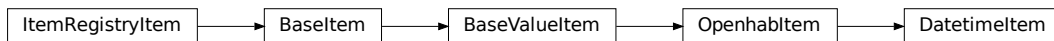
Timestamp of the last time when the item has been updated (read only)

property name: `str`

Returns

Name of the item (read only)

9.2.6 DatetimeItem



class DatetimeItem()

DatetimeItem which accepts and converts the data types from OpenHAB

Variables

- **name** (`str`) – Item name
- **value** (`datetime`) – Current item value (or state in openHAB wording)
- **label** (`Optional[str]`) – Item label or None if not configured
- **tags** (`FrozenSet[str]`) – Item tags
- **groups** (`FrozenSet[str]`) – The groups the item is in
- **metadata** (`Mapping[str, MetaData]`) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

`Any`

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=EventBusListener)`

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value

- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(*value*=<MISSING>)

Send a command to the openHAB item

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

post_value_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value

- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: DateTime**Returns**

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

property `last_update`: `DateTime`

Returns

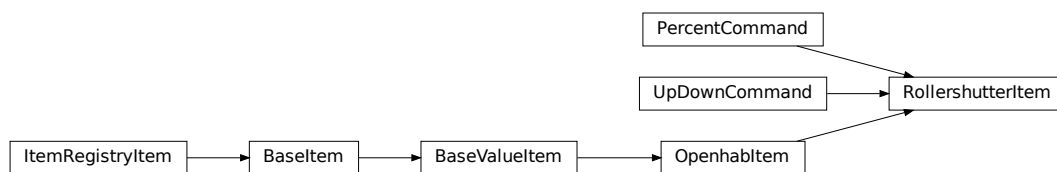
Timestamp of the last time when the item has been updated (read only)

property `name`: `str`

Returns

Name of the item (read only)

9.2.7 RollershutterItem



class RollershutterItem()

RollershutterItem which accepts and converts the data types from OpenHAB

Variables

- **name** (`str`) – Item name
- **value** (`Union[int, float]`) – Current item value (or state in openHAB wording)
- **label** (`Optional[str]`) – Item label or None if not configured
- **tags** (`FrozenSet[str]`) – Item tags
- **groups** (`FrozenSet[str]`) – The groups the item is in
- **metadata** (`Mapping[str, MetaData]`) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

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_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=EventBusListener)`

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(*value=<MISSING>*)

Send a command to the openHAB item

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

post_value_if(*new_value*, *, *equal=<MISSING>*, *eq=<MISSING>*, *not_equal=<MISSING>*,
ne=<MISSING>, *lower_than=<MISSING>*, *lt=<MISSING>*, *lower_equal=<MISSING>*,
le=<MISSING>, *greater_than=<MISSING>*, *gt=<MISSING>*,
greater_equal=<MISSING>, *ge=<MISSING>*, *is=<MISSING>*, *is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: DateTime**Returns**

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

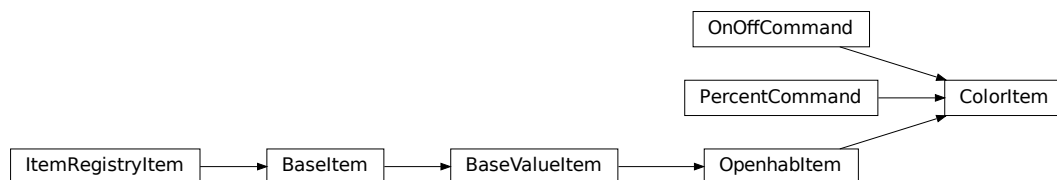
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.8 ColorItem

**class ColorItem()**

ColorItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*Tuple[float, float, float]*) – Current item value (or state in openHAB wording)
- **hue** (*float*) – Hue part of the value
- **saturation** (*float*) – Saturation part of the value
- **brightness** (*float*) – Brightness part of the value

- **label** (*Optional*[*str*]) – Item label or None if not configured
- **tags** (*FrozenSet*[*str*]) – Item tags
- **groups** (*FrozenSet*[*str*]) – The groups the item is in
- **metadata** (*Mapping*[*str*, *MetaData*]) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

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_filter*=None)

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_filter** (Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]) – Event filter. This is typically [ValueUpdateEventFilter](#) or [ValueChangeEventFilter](#) 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. It is also possible to group filters logically with, e.g. [AndFilterGroup](#) and [OrFilterGroup](#)

Return type

TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)

off()

Command item off

oh_post_update(*value*=<MISSING>)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is_*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value

- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(*value=<MISSING>*)

Send a command to the openHAB item

Parameters**value** (*Any*) – (optional) value to be sent. If not specified the current 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 %)

post_value_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post

- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: DateTime**Returns**

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

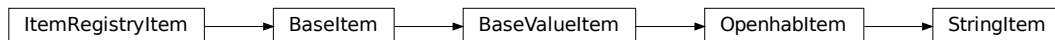
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.9 StringItem

**class StringItem()**

StringItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*str*) – Current item value (or state in openHAB wording)
- **label** (*Optional[str]*) – Item label or None if not configured
- **tags** (*Frozenset[str]*) – Item tags
- **groups** (*Frozenset[str]*) – The groups the item is in
- **metadata** (*Mapping[str, MetaData]*) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (*Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]*) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value*, *, *equal*=<MISSING>, *eq*=<MISSING>, *not_equal*=<MISSING>, *ne*=<MISSING>, *lower_than*=<MISSING>, *lt*=<MISSING>, *lower_equal*=<MISSING>, *le*=<MISSING>, *greater_than*=<MISSING>, *gt*=<MISSING>, *greater_equal*=<MISSING>, *ge*=<MISSING>, *is_*=<MISSING>, *is_not*=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

bool

Returns

True if the new value was posted else *False*

oh_send_command(*value*=<MISSING>)

Send a command to the openHAB item

Parameters

value (*Any*) – (optional) value to be sent. If not specified the current 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

```
post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
               ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>,
               le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>,
               greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else False

```
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: DateTime**Returns**

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

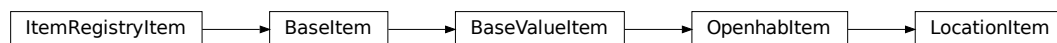
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.10 LocationItem

**class LocationItem()**

LocationItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*Optional[Tuple[float, float, Optional[float]]]*) – Current item value (or state in openHAB wording)

- **label** (*Optional*[*str*]) – Item label or None if not configured
- **tags** (*FrozenSet*[*str*]) – Item tags
- **groups** (*FrozenSet*[*str*]) – The groups the item is in
- **metadata** (*Mapping*[*str*, *MetaData*]) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (*Optional*[*TypeVar*(*HINT_EVENT_FILTER_OBJ*, bound=*EventFilterBase*)] – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(*HINT_EVENT_BUS_LISTENER*, bound=*EventBusListener*)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

```
oh_post_update_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
                  ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>,
                  lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>,
                  gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>,
                  is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

```
oh_send_command(value=<MISSING>)
```

Send a command to the openHAB item

Parameters

value (*Any*) – (optional) value to be sent. If not specified the current 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

```
post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
              ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>,
              le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>,
              greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

```
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: DateTime**Returns**

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

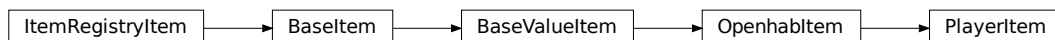
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.11 PlayerItem

**class PlayerItem()**

PlayerItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*str*) – Current item value (or state in openHAB wording)
- **label** (*Optional*[*str*]) – Item label or None if not configured
- **tags** (*FrozenSet*[*str*]) – Item tags
- **groups** (*FrozenSet*[*str*]) – The groups the item is in

- **metadata** (*Mapping*[*str*, *MetaData*]) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback*, *event_filter=None*)

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_filter** (*Optional*[*TypeVar*(*HINT_EVENT_FILTER_OBJ*, bound=*EventFilterBase*)])) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(*HINT_EVENT_BUS_LISTENER*, bound=*EventBusListener*)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value*, *, *equal=<MISSING>*, *eq=<MISSING>*, *not_equal=<MISSING>*, *ne=<MISSING>*, *lower_than=<MISSING>*, *lt=<MISSING>*, *lower_equal=<MISSING>*, *le=<MISSING>*, *greater_than=<MISSING>*, *gt=<MISSING>*, *greater_equal=<MISSING>*, *ge=<MISSING>*, *is_=<MISSING>*, *is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

oh_send_command(*value=<MISSING>*)

Send a command to the openHAB item

Parameters

value (*Any*) – (optional) value to be sent. If not specified the current 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

post_value_if(*new_value*, *, *equal=<MISSING>*, *eq=<MISSING>*, *not_equal=<MISSING>*,
ne=<MISSING>, *lower_than=<MISSING>*, *lt=<MISSING>*, *lower_equal=<MISSING>*,
le=<MISSING>, *greater_than=<MISSING>*, *gt=<MISSING>*,
greater_equal=<MISSING>, *ge=<MISSING>*, *is=<MISSING>*, *is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

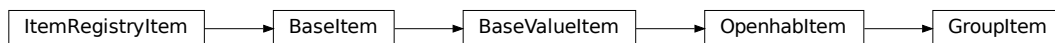
Timestamp of the last time when the item has been updated (read only)

property name: `str`

Returns

Name of the item (read only)

9.2.12 GroupItem

**class GroupItem()**

GroupItem which accepts and converts the data types from OpenHAB

Variables

- **name** (`str`) – Item name
- **value** (`Any`) – Current item value (or state in openHAB wording)
- **label** (`Optional[str]`) – Item label or None if not configured
- **tags** (`FrozenSet[str]`) – Item tags
- **groups** (`FrozenSet[str]`) – The groups the item is in
- **metadata** (`Mapping[str, Metadata]`) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

`Any`

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=EventBusListener)`

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***oh_send_command**(value=<MISSING>)

Send a command to the openHAB item

Parameters**value** (*Any*) – (optional) value to be sent. If not specified the current 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**post_value_if**(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

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

property members: `Tuple[OpenhabItem, ...]`

Resolves and then returns all group members

property name: `str`

Returns

Name of the item (read only)

9.2.13 ImageItem



class ImageItem()

ImageItem which accepts and converts the data types from OpenHAB

Variables

- **name** (`str`) – Item name
- **value** (`bytes`) – Current item value (or state in openHAB wording)
- **image_type** (`Optional[str]`) – image type (e.g. jpg or png)
- **label** (`Optional[str]`) – Item label or None if not configured
- **tags** (`FrozenSet[str]`) – Item tags
- **groups** (`FrozenSet[str]`) – The groups the item is in
- **metadata** (`Mapping[str, MetaData]`) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

`Any`

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound=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_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)*

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

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

```
post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
              ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>,
              le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>,
              greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

```
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: DateTime**Returns**

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

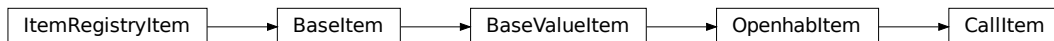
property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

9.2.14 CallItem

**class CallItem()**

CallItem which accepts and converts the data types from OpenHAB

Variables

- **name** (*str*) – Item name
- **value** (*Tuple*[*str*, ...]) – Current item value (or state in openHAB wording)
- **label** (*Optional*[*str*]) – Item label or None if not configured
- **tags** (*FrozenSet*[*str*]) – Item tags
- **groups** (*FrozenSet*[*str*]) – The groups the item is in

- **metadata** (*Mapping[str, MetaData]*) – Item metadata

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. This is a helper function that returns a default in case the item value is None.

Parameters

default_value – Return this value if the item value is None

Return type

Any

Returns

value of the item

listen_event(*callback, event_filter=None*)

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_filter** (*Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]*) – Event filter. This is typically *ValueUpdateEventFilter* or *ValueChangeEventFilter* 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. It is also possible to group filters logically with, e.g. *AndFilterGroup* and *OrFilterGroup*

Return type

TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)

oh_post_update(*value=<MISSING>*)

Post an update to the openHAB item

Parameters

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

oh_post_update_if(*new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

oh_send_command(*value=<MISSING>*)

Send a command to the openHAB item

Parameters

value (*Any*) – (optional) value to be sent. If not specified the current 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

post_value_if(*new_value*, *, *equal=<MISSING>*, *eq=<MISSING>*, *not_equal=<MISSING>*,
ne=<MISSING>, *lower_than=<MISSING>*, *lt=<MISSING>*, *lower_equal=<MISSING>*,
le=<MISSING>, *greater_than=<MISSING>*, *gt=<MISSING>*,
greater_equal=<MISSING>, *ge=<MISSING>*, *is=<MISSING>*, *is_not=<MISSING>*)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else *False*

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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

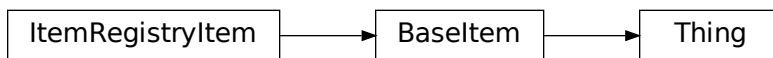
Timestamp of the last time when the item has been updated (read only)

property name: `str`

Returns

Name of the item (read only)

9.2.15 Thing



class Thing(name)

Base class for Things

Variables

- **status** (`ThingStatusEnum`) – Status of the thing (e.g. OFFLINE, ONLINE, ...)
- **status_detail** (`ThingStatusDetailEnum`) – Additional detail for the status
- **status_description** (`str`) – Additional description for the status
- **label** (`str`) – Thing label
- **location** (`str`) – Thing location
- **configuration** (`Mapping[str, Any]`) – Thing configuration
- **properties** (`Mapping[str, Any]`) – Thing properties

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_filter=None*)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound= EventBusListener)`

set_enabled(*enable=True*)

Enable/disable the thing

Parameters

enable (`bool`) – True to enable, False to disable the thing

Returns**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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

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

property name: `str`

Returns

Name of the item (read only)

9.3 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`.

9.3.1 Function parameters

get_thing(*thing*)

Return the complete openHAB thing definition

Parameters

thing (`str` | `ItemRegistryItem`) – name of the thing or the item

Returns

openHAB thing

set_thing_enabled(*thing*, *enabled=True*)

Enable/disable a thing

Parameters

- **thing** (`str` | `ItemRegistryItem`) – name of the thing or the thing object
- **enabled** (`bool`) – True to enable thing, False to disable thing

item_exists(*item*)

Check if an item exists in the openHAB item registry

Parameters

item (`str` | `ItemRegistryItem`) – name of the item or item

Returns

True if item was found

get_item(*item*)

Return the complete openHAB item definition

Parameters

item (`str` | `ItemRegistryItem`) – name of the item or item

Return type

`ItemResp` | `None`

Returns

openHAB item

remove_item(*item*)

Removes an item from the openHAB item registry

Parameters

item (*str* | *ItemRegistryItem*) – name

Returns

True if item was found and removed

create_item(*item_type*, *name*, *label=None*, *category=None*, *tags=None*, *groups=None*, *group_type=None*, *group_function=None*, *group_function_params=None*)

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

Return type

bool

Returns

True if item was created/updated

set_metadata(*item*, *namespace*, *value*, *config*)

Add/set metadata to an item

Parameters

- **item** (*str* | *ItemRegistryItem*) – name of the item or item
- **namespace** (*str*) – namespace, e.g. `stateDescription`
- **value** (*str*) – value
- **config** (*dict*) – configuration e.g. `{"options": "A,B,C"}`

Returns

True if metadata was successfully created/updated

remove_metadata(*item*, *namespace*)

Remove metadata from an item

Parameters

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

Returns

True if metadata was successfully removed

`get_persistence_services()`

Return all available persistence services

`get_persistence_data(item, persistence, start_time, end_time)`

Query historical data from the openHAB persistence service

Parameters

- **item** (`str` | `ItemRegistryItem`) – name of the persistent item
- **persistence** (`str` | `None`) – name of the persistence service (e.g. `rrd4j`, `mapdb`). If not set default will be used
- **start_time** (`datetime` | `None`) – return only items which are newer than this
- **end_time** (`datetime` | `None`) – return only items which are older than this

Return type

`OpenhabPersistenceData`

Returns

last stored data from persistency service

`set_persistence_data(item, persistence, time, state)`

Set a measurement for a item in the persistence service

Parameters

- **item_name** – name of the persistent item
- **persistence** (`str` | `None`) – name of the persistence service (e.g. `rrd4j`, `mapdb`). If not set default will be used
- **time** (`datetime`) – time of measurement
- **state** (`Any`) – state which will be set

Returns

True if data was stored in persistency service

`get_link(item, channel)`

returns the link between an item and a (things) channel

Parameters

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

Return type

`ItemChannelLinkResp`

`remove_link(item, channel)`

removes a link between a (things) channel and an item

Parameters

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

Return type`bool`**Returns**

True on successful removal, otherwise False

create_link(*item*, *channel*, *configuration=None*)

creates a link between an item and a (things) channel

Parameters

- **item** (`str` | `ItemRegistryItem`) – name of the item or item
- **channel** (`str`) – uid of the (things) channel (usually something like AAAA:BBBBB:CCCCC:DDDD:0#SOME_NAME)
- **configuration** (`Optional[dict[str, Any]]`) – optional configuration for the channel

Return type`bool`**Returns**

True on successful creation, otherwise False

send_command(*item*, *command*)

Send the specified command to the item

Parameters

- **item** (`str` | `ItemRegistryItem`) – item name or item
- **command** (`Any`) – command

post_update(*item*, *state*)

Post an update to the item

Parameters

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

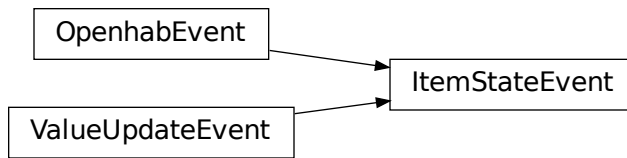
9.4 openHAB event types

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

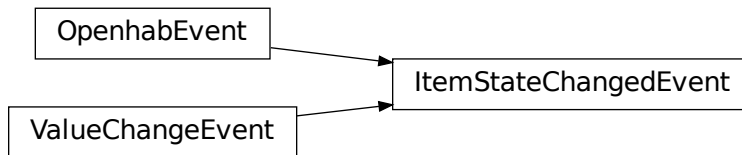
Since this event inherits from `ValueUpdateEvent` you can listen to `ValueUpdateEvent` and it will also trigger for `ItemStateEvent`.



```
class ItemStateEvent(name, value)
```

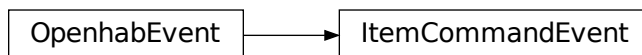
ItemStateChangedEvent

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



```
class ItemStateChangedEvent(name, value, old_value)
```

ItemCommandEvent



```
class ItemCommandEvent(name, value)
```

Variables

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

ItemAddedEvent



```
class ItemAddedEvent(name, type, label, tags, group_names)
```

Variables

- **name** (*str*) –
- **type** (*str*) –
- **label** (*Optional[str]*) –
- **tags** (*FrozenSet[str]*) –
- **groups** (*FrozenSet[str]*) –

ItemUpdatedEvent



```
class ItemUpdatedEvent(name, type, label, tags, group_names)
```

Variables

- **name** (*str*) –
- **type** (*str*) –
- **label** (*Optional[str]*) –
- **tags** (*FrozenSet[str]*) –
- **groups** (*FrozenSet[str]*) –

ItemRemovedEvent

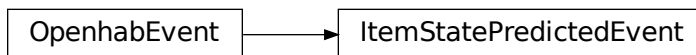


```
class ItemRemovedEvent(name)
```

Variables

- **name** (*str*) –

ItemStatePredictedEvent

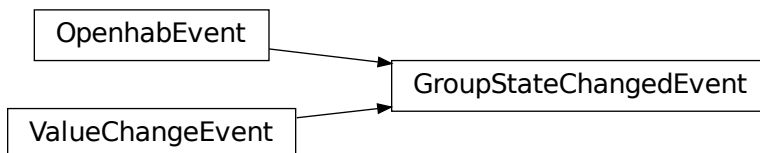


```
class ItemStatePredictedEvent(name, value)
```

Variables

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

GroupStateChangedEvent



```
class GroupStateChangedEvent(name, item, value, old_value)
```

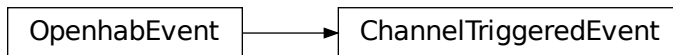
Variables

- **name** (*str*) –

- `item (str)` –
- `value (Any)` –
- `old_value (Any)` –

9.4.2 Channel events

ChannelTriggeredEvent



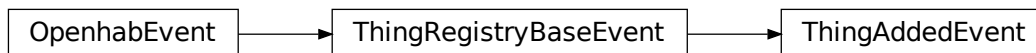
```
class ChannelTriggeredEvent(name="", event="", channel="")
```

Variables

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

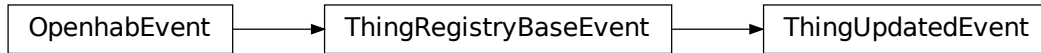
9.4.3 Thing events

ThingAddedEvent



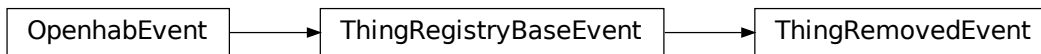
```
class ThingAddedEvent(name, thing_type, label, location, channels, configuration, properties)
```

ThingUpdatedEvent



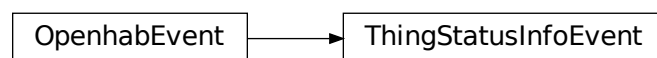
```
class ThingUpdatedEvent(name, thing_type, label, location, channels, configuration, properties)
```

ThingRemovedEvent



```
class ThingRemovedEvent(name, thing_type, label, location, channels, configuration, properties)
```

ThingStatusInfoEvent

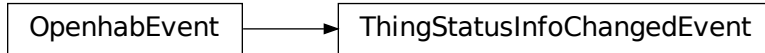


```
class ThingStatusInfoEvent(name="", status=ThingStatusEnum.UNINITIALIZED,
                           detail=ThingStatusDetailEnum.NONE, description="")
```

Variables

- **name** (*str*) –
- **status** (*ThingStatusEnum*) –
- **detail** (*ThingStatusDetailEnum*) –
- **description** (*str*) –

ThingStatusInfoChangedEvent



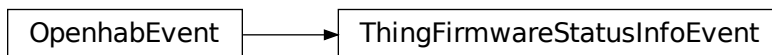
```

class ThingStatusInfoChangedEvent(name="", status=ThingStatusEnum.UNINITIALIZED,
                                   detail=ThingStatusDetailEnum.NONE, description="",
                                   old_status=ThingStatusEnum.UNINITIALIZED,
                                   old_detail=ThingStatusDetailEnum.NONE, old_description="")
  
```

Variables

- **name** (*str*) –
- **status** (*ThingStatusEnum*) –
- **detail** (*ThingStatusDetailEnum*) –
- **description** (*str*) –
- **old_status** (*ThingStatusEnum*) –
- **old_detail** (*ThingStatusDetailEnum*) –
- **old_description** (*str*) –

ThingFirmwareStatusInfoEvent



```

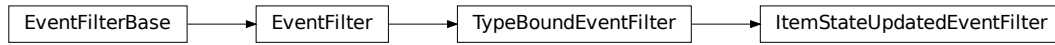
class ThingFirmwareStatusInfoEvent(name="", status="")
  
```

Variables

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

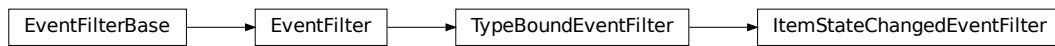
9.4.4 Event filters

ItemStateUpdatedEventFilter



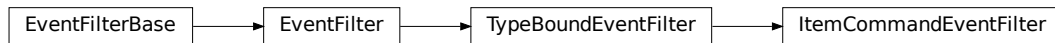
```
class ItemStateUpdatedEventFilter(value=<MISSING>)
```

ItemStateChangedEventFilter



```
class ItemStateChangedEventFilter(value=<MISSING>, old_value=<MISSING>)
```

ItemCommandEventFilter



```
class ItemCommandEventFilter(value=<MISSING>)
```


9.5 Transformations

From openHAB 4 on it's possible to use the existing transformations in HABApp. Transformations are loaded every time when HABApp connects to openHAB. OpenHAB does not issue an event when the transformations change so in order for HABApp to pick up the changes either HABApp or openHAB has to be restarted. Available transformations are logged on connect.

9.5.1 map

The `map transformation` is returned as a dict. If the map transformation is defined with a default the default is used accordingly.

Example:

```
from HABApp.openhab import transformations

TEST_MAP = transformations.map['test.map'] # load the transformation, can be used
↪ anywhere
print(TEST_MAP['test_key'])                # It's a normal dict with keys as str and
↪ values as str

# if all keys or values are numbers they are automatically casted to an int
NUMBERS = transformations.map['numbers.map']
print(NUMBERS[1]) # Note that the key is an int
```

```
test_value
test number meaning
```

9.6 Textual thing configuration

9.6.1 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.

9.6.2 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 `.yaml` 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.

9.6.3 File Structure

Configuration is done through a `.yaml` 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

# 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,
```

(continues on next page)

(continued from previous page)

```

    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 they have to be applied multiple times to the same field.

```

- 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
    ...

```

9.6.4 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
```

9.6.5 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 under 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 ] }
```

9.6.6 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`

9.6.7 Example

Log output

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

```
Loading /config/thing_philio.yml!
```

```
+-----  
↔-----
```

(continues on next page)

(continued from previous page)

```

↪---+
|
↪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
↪    |
| 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 |
↪                               |

```

(continues on next page)

(continued from previous page)

```

+-----+-----+-----+-----+
↪-----+-----+
|      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']    |                    |                    |
| 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

```

(continues on next page)

(continued from previous page)

Would set {6: 4} for zwave:device:controller:node5

```

+-----+
|                                     Channels for zwave:philio_pst02a_00_000                                     |
+-----+
|                                     |
+-----+
|                                     |
+-----+
| channel_label | channel_kind | channel_uid | channel_type |
+-----+
| 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!

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!

```

(continues on next page)

(continued from previous page)

```

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')

```

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>

(continues on next page)

(continued from previous page)

```

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>
↳
↳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>
↳
↳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>
↳
↳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>
↳
↳temperature"}

```

9.7 Example openHAB rules

9.7.1 Example 1

```

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

class MyOpenhabRule(HABApp.Rule):

    def __init__(self):
        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

```

(continues on next page)

(continued from previous page)

```

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()

9.7.2 Check status of things

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

```

from HABApp import Rule
from HABApp.core.events import EventFilter
from HABApp.openhab.events import ThingStatusInfoChangedEvent
from HABApp.openhab.items import Thing

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

        for thing in self.get_items(Thing):
            thing.listen_event(self.thing_status_changed,
                               EventFilter(ThingStatusInfoChangedEvent))
            print(f'{thing.name}: {thing.status}')

```

(continues on next page)

(continued from previous page)

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

```
CheckAllThings()
```

9.7.3 Check status if thing is constant

Sometimes Things recover automatically from small outages. This rule only triggers when 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)
        watcher.listen_event(self.thing_no_change)

    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
```


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. Please note that subscriptions made this way are volatile, and will only remain until the next disconnect. For persistent subscriptions use the corresponding entry in the configuration file. By default HABApp listens to all topics so the topics can be used in `listen_event`.

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.

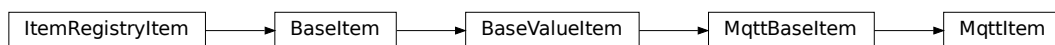
```
from HABApp.mqtt.items import MqttItem
from HABApp.core.events import ValueChangeEvent

# Messages with a retain flag will automatically create a corresponding item in HABApp.
# All other items have to be created manually
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 MqttItem()

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. This is a helper function that returns a default in case the item value is None.

Parameters**default_value** – Return this value if the item value is None**Return type**`Any`**Returns**

value of the item

listen_event(`callback, event_filter=None`)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type`TypeVar(HINT_EVENT_BUS_LISTENER, bound= 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

post_value_if(`new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>, ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>, le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>, greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)`)

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post

- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type`bool`**Returns***True* if the new value was posted else *False***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.

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: DateTime**Returns**

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

property last_update: DateTime**Returns**

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

property name: str**Returns**

Name of the item (read only)

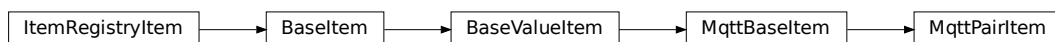
10.3.2 MqttPairItem

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 `MqttPairItem()`

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. This is a helper function that returns a default in case the item value is `None`.

Parameters

default_value – Return this value if the item value is `None`

Return type

`Any`

Returns

value of the item

listen_event(`callback, event_filter=None`)

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_filter** (`Optional[TypeVar(HINT_EVENT_FILTER_OBJ, bound=EventFilterBase)]`) – Event filter. This is typically `ValueUpdateEventFilter` or `ValueChangeEventFilter` 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. It is also possible to group filters logically with, e.g. `AndFilterGroup` and `OrFilterGroup`

Return type

`TypeVar(HINT_EVENT_BUS_LISTENER, bound= 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

```
post_value_if(new_value, *, equal=<MISSING>, eq=<MISSING>, not_equal=<MISSING>,
               ne=<MISSING>, lower_than=<MISSING>, lt=<MISSING>, lower_equal=<MISSING>,
               le=<MISSING>, greater_than=<MISSING>, gt=<MISSING>,
               greater_equal=<MISSING>, ge=<MISSING>, is_=<MISSING>, is_not=<MISSING>)
```

Post a value depending on the current state of the item. If one of the comparisons is true the new state will be posted.

Parameters

- **new_value** – new value to post
- **equal** – item state has to be equal to the passed value
- **eq** – item state has to be equal to the passed value
- **not_equal** – item state has to be not equal to the passed value
- **ne** – item state has to be not equal to the passed value
- **lower_than** – item state has to be lower than the passed value
- **lt** – item state has to be lower than the passed value
- **lower_equal** – item state has to be lower equal the passed value
- **le** – item state has to be lower equal the passed value
- **greater_than** – item state has to be greater than the passed value
- **gt** – item state has to be greater than the passed value
- **greater_equal** – item state has to be greater equal the passed value
- **ge** – item state has to be greater equal the passed value
- **is** – item state has to be the same object as the passt value (e.g. None)
- **is_not** – item state has to be not the same object as the passt value (e.g. None)

Return type

`bool`

Returns

True if the new value was posted else False

```
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: `DateTime`

Returns

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

property last_update: `DateTime`

Returns

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

property name: `str`

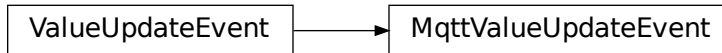
Returns

Name of the item (read only)

10.4 Mqtt event types

10.4.1 MqttValueUpdateEvent

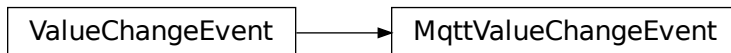
Since this event inherits from *ValueUpdateEvent* you can listen to *ValueUpdateEvent* and it will also trigger for *MqttValueUpdateEvent*.



```
class MqttValueUpdateEvent(name, value)
```

10.4.2 MqttValueChangeEvent

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



```
class MqttValueChangeEvent(name, value, old_value)
```

10.5 Example MQTT rule

```
import datetime
import random

import HABApp
from HABApp.core.events import ValueUpdateEvent, ValueUpdateEventFilter
from HABApp.mqtt.items import MqttItem

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

        self.run.every(
            start_time=datetime.timedelta(seconds=10),
```

(continues on next page)

(continued from previous page)

```
        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, ValueUpdateEventFilter())

    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()
```


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.F	The corresponding events trigger a load/unload of the file specified in the event	<i>RequestFileLoadEvent</i> and <i>RequestFileUnloadEvent</i>
HABApp.I	All infos in functions and rules of HABApp create an according event	<i>str</i>
HABApp.W	All warnings in functions (e.g. caught exceptions) and rules of HABApp create an according event	<i>HABAppException</i> or <i>str</i>
HABApp.E	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).	<i>HABAppException</i> or <i>str</i>

class RequestFileLoadEvent(*name*)

Request (re-) loading of the specified file

Variables

filename (*str*) – relative filename

class RequestFileUnloadEvent(*name*)

Request unloading of the specified file

Variables

filename (*str*) – relative filename

class 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. The 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 Running Python code on startup

It's possible to run arbitrary code during the startup of HABApp. This can be achieved by creating a module/package called HABAppUser. HABApp will try to import it before loading the configuration and thus execute the code. The module/package must be importable so it has to be in one of the PATH/PYTHONPATH folders or in the current working directory.

11.4 Invoking openHAB actions

The openHAB REST interface does not expose [actions](#), and thus there is no way to trigger them from HABApp. Even if it is not possible to create an 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 a couple of items to accept values from HABApp, and place them in `/etc/openhab2/items/habapp-bridge.items`:

```
String AudioVoiceSinkName

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 onAudioStreamURLChanged(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 onAudioFileLocationChanged(event):
    file_location = scope.items[event.itemName].toString()
```

(continues on next page)

(continued from previous page)

```

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_sync.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface_sync.send_command(ACTION_AUDIO_LOCAL_FILE_LOCATION_ITEM_
↳NAME, file_location)

def play_stream_url(sink_name: str, url: str):
    """ Plays a stream URL on the given audio sink. """
    HABApp.openhab.interface_sync.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface_sync.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_sync.send_command(ACTION_AUDIO_SINK_ITEM_NAME, sink_name)
    HABApp.openhab.interface_sync.send_command(ACTION_TEXT_TO_SPEECH_MESSAGE_ITEM_NAME,
↳tts)

```

11.5 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()`, `SwitchItem.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
```


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

delete(*url*, *params=None*, ***kwargs*)

http delete 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](#)
- **kwargs** (*Any*) – See [aiohttp request](#) for further possible kwargs

Return type

`_RequestContextManager`

Returns

awaitable

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](#)
- **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

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

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)
        async with self.async_http.get('http://httpbin.org/get') as resp:
            print(resp)
            print(await resp.text())

AsyncRule()
```


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))
```

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))
```

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))
```

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))
```

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 Rate limiter

A simple rate limiter implementation which can be used in rules. The limiter is not rule bound so the same limiter can be used in multiples files. It also works as expected across rule reloads.

13.2.1 Defining limits

Limits can either be explicitly added or through a textual description. If the limit does already exist it will not be added again. It's possible to explicitly create the limits or through some small textual description with the following syntax:

```
[count] [per|in|/] [count (optional)] [s|sec|second|m|min|minute|hour|h|day|month|year].  
↪[s (optional)]
```

Whitespaces are ignored and can be added as desired

Examples:

- 5 per minute
- 20 in 15 mins
- 300 / hour

13.2.2 Fixed window elastic expiry algorithm

This algorithm implements a fixed window with elastic expiry. That means if the limit is hit the interval time will be increased by the expiry time.

For example 3 per minute:

- First hit comes 00:00:00. Two more hits at 00:00:59. All three pass, intervall goes from 00:00:00 - 00:01:00. Another hit comes at 00:01:01 and passes. The intervall now goes from 00:01:01 - 00:02:01.
- First hit comes 00:00:00. Two more hits at 00:00:30. All three pass. Another hit comes at 00:00:45, which gets rejected and the intervall now goes from 00:00:00 - 00:01:45. A rejected hit makes the interval time longer by expiry time. If another hit comes at 00:01:30 it will also get rejected and the intervall now goes from 00:00:00 - 00:02:30.

13.2.3 Leaky bucket algorithm

The leaky bucket algorithm is based on the analogy of a bucket that leaks at a constant rate. As long as the bucket is not full the hits will pass. If the bucket overflows the hits will get rejected. Since the bucket leaks at a constant rate it will gradually get empty again thus allowing hits to pass again.

13.2.4 Example

```
from HABApp.util import RateLimiter

# Create or get existing, name is case insensitive
limiter = RateLimiter('MyRateLimiterName')

# define limits, duplicate limits of the same algorithm will only be added once
# These lines all define the same limit so it'll result in only one limiter added
limiter.add_limit(5, 60) # add limits explicitly
limiter.parse_limits('5 per minute').parse_limits('5 in 60s', '5/60seconds') # add
↳ limits through text

# add additional limit with leaky bucket algorithm
limiter.add_limit(10, 100, algorithm='leaky_bucket')

# add additional limit with fixed window elastic expiry algorithm
limiter.add_limit(10, 100, algorithm='fixed_window_elastic_expiry')

# Test the limit without increasing the hits
for _ in range(100):
    assert limiter.test_allow()

# the limiter will allow 5 calls ...
for _ in range(5):
    assert limiter.allow()

# and reject the 6th
assert not limiter.allow()

# It's possible to get statistics about the limiter and the corresponding windows
print(limiter.info())

# There is a counter that keeps track of the total skips that can be reset
print('Counter:')
print(limiter.total_skips)
limiter.reset() # Can be reset
print(limiter.total_skips)
```

```
LimiterInfo(skips=1, total_skips=1, limits=[LeakyBucketLimitInfo(hits=5, skips=1,
↳ limit=5, time_remaining=11.999655897999219), LeakyBucketLimitInfo(hits=5, skips=0,
↳ limit=5, time_remaining=9.999655897999393), FixedWindowElasticExpiryLimitInfo(hits=5,
↳ skips=0, limit=10, time_remaining=99.99997635700038)])
Counter:
1
0
```

13.2.5 Recommendation

Limiting external requests to an external API works well with the leaky bucket algorithm (maybe with some initial hits). For limiting notifications the best results can be achieved by combining both algorithms. Fixed window elastic expiry will notify but block until an issue is resolved, that's why it's more suited for small intervals. Leaky bucket will allow hits even while the issue persists, that's why it's more suited for larger intervals.

```
from HABApp.util import RateLimiter

limiter = RateLimiter('MyNotifications')
limiter.parse_limits('5 in 1 minute', algorithm='fixed_window_elastic_expiry')
limiter.parse_limits("20 in 1 hour", algorithm='leaky_bucket')
```

13.2.6 Documentation

RateLimiter(*name*)

Create a new rate limiter or return an already existing one with a given name.

Parameters

name (*str*) – case insensitive name of limiter

Return type

Limiter

Returns

Rate limiter object

class Limiter(*name*)

property total_skips: *int*

A counter to track skips which can be manually reset

add_limit(*allowed, interval, *, initial_hits=0, algorithm='leaky_bucket'*)

Add a new rate limit

Parameters

- **allowed** (*int*) – How many hits are allowed
- **interval** (*int*) – Interval in seconds
- **initial_hits** (*int*) – How many hits the limit already has when it gets initially created
- **algorithm** (*Literal*['leaky_bucket', 'fixed_window_elastic_expiry']) – Which algorithm should this limit use

Return type

Limiter

parse_limits(**text, initial_hits=0, algorithm='leaky_bucket'*)

Add one or more limits in textual form, e.g. 5 in 60s, 10 per hour or 10/15 mins. If the limit does already exist it will not be added again.

Parameters

- **text** (*str*) – textual description of limit
- **initial_hits** (*int*) – How many hits the limit already has when it gets initially created

- **algorithm** (`Literal['leaky_bucket', 'fixed_window_elastic_expiry']`) – Which algorithm should these limits use

Return type

Limiter

allow()

Test the limit(s).

Return type

`bool`

Returns

True if allowed, False if forbidden

test_allow()

Test the limit(s) without hitting it. Calling this will not increase the hit counter.

Return type

`bool`

Returns

True if allowed, False if forbidden

info()

Get some info about the limiter and the defined windows

Return type

LimiterInfo

reset()

Reset the skip counter

Return type

Limiter

class LimiterInfo(*skips, total_skips, limits*)

skips: `int`

How many entries were skipped in the active interval(s)

total_skips: `int`

How many entries were skipped in total

limits: `List[Union[FixedWindowElasticExpiryLimitInfo, LeakyBucketLimitInfo]]`

Info for every limit

class FixedWindowElasticExpiryLimitInfo(*hits, skips, limit, time_remaining*)

time_remaining: `float`

Time remaining until this window will reset

hits: `int`

Hits

skips: `int`

Skips

limit: `int`

Boundary


```
class LeakyBucketLimitInfo(hits, skips, limit, time_remaining)
```

```
    time_remaining: float
```

Time remaining until the next drop

```
    hits: int
```

Hits

```
    skips: int
```

Skips

```
    limit: int
```

Boundary

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 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

```
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 Fade

Fade is a helper class which allows to easily fade a value up or down.

13.4.1 Example

This example shows how to fade a Dimmer from 0 to 100 in 30 secs

```
from HABApp import Rule
from HABApp.openhab.items import DimmerItem
from HABApp.util import Fade

class FadeExample(Rule):
    def __init__(self):
        super().__init__()
        self.dimmer = DimmerItem.get_item('Dimmer1')
        self.fade = Fade(callback=self.fade_value) # self.dimmer.percent would also be a good callback in this example

        # Setup the fade and schedule its execution
        # Fade from 0 to 100 in 30s
        self.fade.setup(0, 100, 30).schedule_fade()

    def fade_value(self, value):
        self.dimmer.percent(value)

FadeExample()
```

This example shows how to fade three values together (e.g. for an RGB strip)

```
from HABApp import Rule
from HABApp.openhab.items import DimmerItem
from HABApp.util import Fade

class Fade3Example(Rule):
    def __init__(self):
        super().__init__()
        self.fade1 = Fade(callback=self.fade_value)
        self.fade2 = Fade()
        self.fade3 = Fade()

        # Setup the fades and schedule the execution of one fade where the value gets updated every sec
        self.fade3.setup(0, 100, 30)
        self.fade2.setup(0, 50, 30)
        self.fade1.setup(0, 25, 30, min_step_duration=1).schedule_fade()

    def fade_value(self, value):
        value1 = value
        value2 = self.fade2.get_value()
        value3 = self.fade3.get_value()
```

(continues on next page)

(continued from previous page)

```
Fade3Example()
```

13.4.2 Documentation

class Fade(*callback=None, min_value=0, max_value=100*)

Helper to easily fade values up/down

Variables

- **min_value** – minimum valid value for the fade value
- **max_value** – maximum valid value for the fade value
- **callback** – Function with one argument that will be automatically called with the new values when the scheduled fade runs

setup(*start_value, stop_value, duration, min_step_duration=0.2, now=None*)

Calculates everything that is needed to fade a value

Parameters

- **start_value** (*Union[int, float]*) – Start value
- **stop_value** (*Union[int, float]*) – Stop value
- **duration** (*Union[int, float, timedelta]*) – How long shall the fade take
- **min_step_duration** (*float*) – minimum step duration (min 0.2 secs)
- **now** (*Optional[float]*) – time.time() timestamp to sync multiple fades together

Return type

Fade

get_value(*now=None*)

Returns the current value. If the fade is finished it will always return the stop value.

Parameters

now (*Optional[float]*) – time.time() timestamp for which the value shall be returned. Can be used to sync multiple fades together. Not required.

Return type

float

Returns

current value

property is_finished: *bool*

True if the fade is finished

schedule_fade()

Automatically run the fade with the Scheduler. The callback can be used to set the current fade value e.g. on an item. Calling this on a running fade will restart the fade

Return type

Fade

stop_fade()

Stop the scheduled fade. This can be called multiple times without error

13.5 EventListenerGroup

EventListenerGroup is a helper class which allows to subscribe to multiple items at once. All subscriptions can be canceled together, too. This is useful if e.g. something has to be done once after a sensor reports a value.

13.5.1 Example

This is a rule which will turn on the lights once (!) in a room on the first movement in the morning. The lights will only turn on after 4 and before 8 and two movement sensors are used to pick up movement.

```
from datetime import time

from HABApp import Rule
from HABApp.core.events import ValueChangeEventFilter
from HABApp.openhab.items import SwitchItem, NumberItem
from HABApp.util import EventListenerGroup

class EventListenerGroupExample(Rule):
    def __init__(self):
        super().__init__()
        self.lights = SwitchItem.get_item('RoomLights')
        self.sensor_move_1 = NumberItem.get_item('MovementSensor1')
        self.sensor_move_2 = NumberItem.get_item('MovementSensor2')

        # use a list of items which will be subscribed with the same callback and event
        self.listeners = EventListenerGroup().add_listener(
            [self.sensor_move_1, self.sensor_move_2], self.sensor_changed,
            ValueChangeEventFilter())

        self.run.on_every_day(time(4), self.listen_sensors)
        self.run.on_every_day(time(8), self.sensors_cancel)

    def listen_sensors(self):
        self.listeners.listen()

    def sensors_cancel(self):
        self.listeners.cancel()

    def sensor_changed(self, event):
        self.listeners.cancel()
        self.lights.on()

EventListenerGroupExample()
```

13.5.2 Documentation

class `EventListenerGroup`

Helper to create/cancel multiple event listeners simultaneously

property **active**: `bool`

Returns

True if the listeners are currently active

`listen()`

Create all event listeners. If the event listeners are already active this will do nothing.

`cancel()`

Cancel the active event listeners. If the event listeners are not active this will do nothing.

`activate_listener(name)`

Resume a previously deactivated listener creator in the group.

Parameters

name (`str`) – item name or alias of the listener

Returns

True if it was activated, False if it was already active

`deactivate_listener(name, cancel_if_active=True)`

Exempt the listener creator from further listener/cancel calls

Parameters

- **name** (`str`) – item name or alias of the listener
- **cancel_if_active** – Cancel the listener if it is active

Returns

True if it was deactivated, False if it was already deactivated

`add_listener(item, callback, event_filter, alias=None)`

Add an event listener to the group

Parameters

- **item** (`Union[TypeVar(HINT_ITEM_OBJ, bound= BaseItem), Iterable[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)]]`) – Single or multiple items
- **callback** (`Callable[[Any], Any]`) – Callback for the item(s)
- **event_filter** (`TypeVar(HINT_EVENT_FILTER_OBJ, bound= EventFilterBase)`) – Event filter for the item(s)
- **alias** (`Optional[str]`) – Alias if an item with the same name does already exist (e.g. if different callbacks shall be created for the same item)

Return type

`EventListenerGroup`

Returns

self

add_no_update_watcher(*item*, *callback*, *seconds*, *alias=None*)

Add an no update watcher to the group. On listen this will create a no update watcher and the corresponding event listener that will trigger the callback

Parameters

- **item** (`Union[TypeVar(HINT_ITEM_OBJ, bound= BaseItem), Iterable[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)]]`) – Single or multiple items
- **callback** (`Callable[[Any], Any]`) – Callback for the item(s)
- **seconds** (`Union[int, float, timedelta]`) – No update time for the no update watcher
- **alias** (`Optional[str]`) – Alias if an item with the same name does already exist (e.g. if different callbacks shall be created for the same item)

Return type

EventListenerGroup

Returns

self

add_no_change_watcher(*item*, *callback*, *seconds*, *alias=None*)

Add a no change watcher to the group. On listen this will create a no change watcher and the corresponding event listener that will trigger the callback

Parameters

- **item** (`Union[TypeVar(HINT_ITEM_OBJ, bound= BaseItem), Iterable[TypeVar(HINT_ITEM_OBJ, bound= BaseItem)]]`) – Single or multiple items
- **callback** (`Callable[[Any], Any]`) – Callback for the item(s)
- **seconds** (`Union[int, float, timedelta]`) – No update time for the no change watcher
- **alias** (`Optional[str]`) – Alias if an item with the same name does already exist (e.g. if different callbacks shall be created for the same item)

Return type

EventListenerGroup

Returns

self

13.6 MultiModelItem

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.

13.6.1 Basic Example

```
import HABApp
from HABApp.core.events import ValueUpdateEventFilter
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, ValueUpdateEventFilter())

        # 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)
        # Add the auto mode with priority 0 and the manual mode with priority 10
        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 shown 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
```

13.6.2 Advanced Example

```
import logging
import HABApp
from HABApp.core.events import ValueUpdateEventFilter
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, ValueUpdateEventFilter())

        # helper to print the heading so we have a nice output
        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 achieved by automatically
        ↪ disabling 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

```

13.6.3 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.openhab.items import SwitchItem
from HABApp.util.multimode import MultiModeItem, SwitchItemValueMode, ValueMode

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

```

(continues on next page)

(continued from previous page)

```

# 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>

```

13.6.4 Documentation

MultiModeItem

class MultiModeItem()

Prioritizer *Item*

classmethod get_create_item(name, initial_value=None, default_value=<MISSING>)

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
- **default_value** – Default value that will be sent if no mode is active

Return type

MultiModeItem

Returns

The created or existing 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 nothing 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** (*TypeVar*(HINT_BASE_MODE, bound= 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*, *TypeVar*(HINT_BASE_MODE, bound= 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

TypeVar(HINT_BASE_MODE, bound= BaseMode)

Returns

The requested MultiModeValue

calculate_value()

Recalculate the value. If the new value is not MISSING the calculated value will be set as the item state and the corresponding events will be generated.

Return type

Any

Returns

new value

ValueMode

```
class 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*) – Timestamp of the last update/enable of this value
- **auto_disable_after** (*Optional[timedelta]*) – Automatically disable this mode after a given timedelta on the next recalculation
- **auto_disable_func** (*Optional[Callable[[Any, 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** (*Optional[Callable[[Any, Any], 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.

property value

Returns the current value

property enabled: bool

Returns if the value is enabled

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

Returns

False if the value was not set, True otherwise

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

Return type

bool

Returns

True if the value was set else False

cancel()

Remove the mode from the parent `MultiModeItem` and stop processing it

SwitchItemValueMode

```
class 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*) – Timestamp of the last update/enable of this value
- **auto_disable_after** (*Optional[timedelta]*) – Automatically disable this mode after a given timedelta on the next recalculation
- **auto_disable_func** (*Optional[Callable[[Any, 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** (*Optional[Callable[[Any, Any], 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.

cancel()

Remove the mode from the parent MultiModeItem and stop processing it

property enabled: bool

Returns if the value is enabled

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

Returns

False if the value was not set, True otherwise

property value

Returns the current value

ADDITIONAL RULE EXAMPLES

14.1 Using the scheduler

```
from datetime import datetime, time, timedelta

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, ItemStateUpdatedEventFilter
from HABApp.openhab.items import OpenhabItem

class ExampleOpenhabToMQTTRule(HABApp.Rule):
    """This Rule mirrors all updates from OpenHAB to MQTT"""

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

        for item in self.get_items(OpenhabItem):
            item.listen_event(self.process_update, ItemStateUpdatedEventFilter())

    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, EventFilter

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

        my_item = Item.get_item('test_watch')

        # Create an event when the item doesn't change for 5 secs and
        # create a watcher for ItemNoChangeEvent with 5s const time
        my_item.watch_change(5).listen_event(self.item_constant_5s)

        # Just create an event when the item doesn't change for 10 secs
        my_item.watch_change(10)

        # Listen to all ItemNoChangeEvents for the item
        my_item.listen_event(self.item_constant, EventFilter(ItemNoChangeEvent))

        # Set the item to a value to generate the ItemNoChangeEvent events
```

(continues on next page)

(continued from previous page)

```

my_item.set_value('my_value')

def item_constant_5s(self, event):
    print(f'Item 5s const: {event}')

def item_constant(self, event):
    print(f'Item const: {event}')

MyRule()

```

```

Item 5s const: <ItemNoChangeEvent name: test_watch, seconds: 5>
Item const: <ItemNoChangeEvent name: test_watch, seconds: 5>
Item const: <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, ValueUpdateEventFilter

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, ValueUpdateEventFilter())

        self.movement2 = Item.get_item('movement_sensor2')
        self.movement2.listen_event(self.movement, ValueUpdateEventFilter())

    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, use a notification service to send the error message to the mobile device or send an email with the error message. See *Advanced Usage* for more information about the available internal topics. It also uses the built in *rate limiter* to limit the amount of notifications.

```
import HABApp
from HABApp.core.events.habapp_events import HABAppException
from HABApp.core.events import EventFilter
from HABApp.util import RateLimiter

# Set up rate limiter to limit the amount of notifications
LIMITER = RateLimiter('MyNotifications')
LIMITER.parse_limits('5 in 1 minute', algorithm='fixed_window_elastic_expiry')
LIMITER.parse_limits('20 in 1 hour', algorithm='leaky_bucket')

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

        # Listen to all errors
        self.listen_event('HABApp.Errors', self.on_error, EventFilter(HABAppException))

    def on_error(self, error_event: HABAppException):
        msg = error_event.to_str() if isinstance(error_event, HABAppException) else
        ↪error_event

        # use limiter
        if not LIMITER.allow():
            return None

        # Replace this part with your notification logic
        print('Error in rules:')
        print(msg)

NotifyOnError()

# this is a faulty rule as an 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()
```

Error in rules:

(continues on next page)

(continued from previous page)

```
Exception in TestRule.FaultyRule.faulty_function: division by zero
File "<string>", line 46 in faulty_function
```

```
-----
Traceback (most recent call last):
```

```
  File "/home/docs/checkouts/readthedocs.org/user_builds/habapp/checkouts/develop/src/
↳ HABApp/core/internals/wrapped_function/wrapped_thread.py", line 94, in run
    self.func_obj(*self.func_args, **self.func_kwargs)
  File "<string>", line 46, in faulty_function
ZeroDivisionError: division by zero
```


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 it can be easily tested: Set `autoupdate` to off, then watch the item state after sending a command to it. 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*.

TROUBLESHOOTING

16.1 Warnings

16.1.1 Starting of <FUNC_NAME> took too long.

This warning appears in the HABApp log, e.g.:

```
Starting of MyRule.my_func took too long: 0.08s. Maybe there are not enough threads?
```

It means that the duration from when the event was received to the start of the execution of the function took longer than expected.

This can be the case if suddenly many events are received at once. Another reason for this warning might be that currently running function calls take too long to finish and thus no free workers are available. This can either be the case for complex calculations, but most of the time it's blocking function calls or a `time.sleep` call.

If these warnings pile up in the log it's an indicator that the worker is congested. Make sure there is no use of long sleeps and instead the scheduler is used.

If this warning only appears now and then it can be ignored.

16.1.2 Execution of <FUNC_NAME> took too long

This warning appears in the HABApp log, e.g.:

```
Execution of MyRule.my_long_func took too long: 15.25s
```

It means that the function took very long to execute. By default HABApp has 10 threads and each function call will happen in one of those threads. Normally this is not a problem because functions finish rather quickly and the used thread is free for the next function call. When functions take very long to execute and multiple of these functions run parallel it's possible that all threads are blocked. HABApp will then appear to “hang” and can not process new events.

If the function uses `time.sleep` it can be split up and the scheduler can be used instead.

Long running scripts (>10s) which do not interact with openHAB can be run as a separate process with `execute_python()`. The script can e.g. print the result as a json which HABApp can read and load again into the proper data structures.

If this warning only appears now and then it can be ignored.

16.1.3 Item <ITEM_NAME> is a UoM item but “unit” is not found in item metadata

Starting from OH4 it's possible to use an internal normalisation unit and scale for UoM items. To use this normalisation one has to set the `unit` metadata on the item.:

```
Number:Temperature  My_Temp      { unit="°C" }
```

It's strongly recommend to explicitly set this normalisation value. Only when used it'll prevent graphs and persisted values from changing the unit and scale which would result in broken graphs or broken persisted data.

16.2 Errors

16.2.1 ValueError: Line is too long

The underlying libraries of HABApp use a buffer to process each request and event from openHAB. If the openHAB items contain images this buffer might be not enough and a `ValueError: Line is too long` error will appear in the logs. See [the openHAB connection options](#) on how to increase the buffer. The maximum image size that can be used without error is ~40% of the buffer size.

CLASS REFERENCE

Reference for returned classes from some functions. These are not intended to be created by the user.

17.1 Watches

17.1.1 ItemNoUpdateWatch

class `ItemNoUpdateWatch`(*name*, *secs*)

EVENT

alias of *ItemNoUpdateEvent*

cancel()

Cancel the item watch

listen_event(*callback*)

Listen to (only) the event that is emitted by this watcher

17.1.2 ItemNoChangeWatch

class `ItemNoChangeWatch`(*name*, *secs*)

EVENT

alias of *ItemNoChangeEvent*

cancel()

Cancel the item watch

listen_event(*callback*)

Listen to (only) the event that is emitted by this watcher

17.2 Scheduler

17.2.1 OneTimeJob

class `OneTimeJob`(*parent*, *func*)

cancel()

Cancel the job.

get_next_run()

Return the next execution timestamp.

Return type

`datetime`

remaining()

Returns the remaining time to the next run or None if the job is not scheduled

Return type

`Optional[timedelta]`

Returns

remaining time as a timedelta or None

to_item(*item*)

Sends the next execution (date)time to an item. Sends None if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str` | `BaseValueItem` | `None`) – item name or item, None to disable

17.2.2 CountdownJob

class `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`

remaining()

Returns the remaining time to the next run or None if the job is not scheduled

Return type

`Optional[timedelta]`

Returns

remaining time as a timedelta or None

stop()

Stops the countdown so it can be started again with a call to reset

to_item(item)

Sends the next execution (date)time to an item. Sends None if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str` | `BaseValueItem` | `None`) – item name or item, None to disable

17.2.3 ReoccurringJob

class `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`

remaining()

Returns the remaining time to the next run or *None* if the job is not scheduled

Return type

`Optional[timedelta]`

Returns

remaining time as a `timedelta` or *None*

to_item(*item*)

Sends the next execution (date)time to an item. Sends *None* if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str | BaseValueItem | None`) – item name or item, *None* to disable

17.2.4 DayOfWeekJob

class `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

remaining()Returns the remaining time to the next run or *None* if the job is not scheduled**Return type***Optional*[*timedelta*]**Returns**remaining time as a *timedelta* or *None*

time(*time*)

Set a time of day when the job will run.

Parameters

time (`Union[time, datetime]`) – time

Return type

DayOfWeekJob

to_item(*item*)

Sends the next execution (date)time to an item. Sends `None` if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str | BaseValueItem | None`) – item name or item, `None` to disable

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

17.2.5 DawnJob

class `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`

remaining()

Returns the remaining time to the next run or *None* if the job is not scheduled

Return type
`Optional[timedelta]`

Returns
 remaining time as a `timedelta` or *None*

to_item(*item*)

Sends the next execution (date)time to an item. Sends *None* if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str | BaseValueItem | None`) – item name or item, *None* to disable

17.2.6 SunriseJob

class 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`

remaining()

Returns the remaining time to the next run or `None` if the job is not scheduled

Return type

`Optional[timedelta]`

Returns

remaining time as a timedelta or None

to_item(*item*)

Sends the next execution (date)time to an item. Sends None if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (*str* | *BaseValueItem* | *None*) – item name or item, None to disable

17.2.7 SunsetJob

class *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`

remaining()

Returns the remaining time to the next run or `None` if the job is not scheduled

Return type

`Optional[timedelta]`

Returns

remaining time as a `timedelta` or `None`

to_item(*item*)

Sends the next execution (date)time to an item. Sends `None` if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters

item (`str` | `BaseValueItem` | `None`) – item name or item, `None` to disable

17.2.8 DuskJob

class 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

remaining()

Returns the remaining time to the next run or None if the job is not scheduled

Return type

Optional[timedelta]

Returns

remaining time as a timedelta or None

to_item(item)

Sends the next execution (date)time to an item. Sends None if the job is not scheduled. Every time the scheduler updates to a new (date)time the item will also receive the updated time.

Parameters**item** (str | BaseValueItem | None) – item name or item, None to disable

INDICES AND TABLES

- `genindex`
- `modindex`

PYTHON MODULE INDEX

h

`HABApp.openhab.interface_sync`, [128](#)
`HABApp.rule.interfaces.http`, [169](#)
`HABApp.util`, [173](#)

A

activate_listener() (*EventListenerGroup* method), 183
 active (*EventListenerGroup* property), 183
 add_limit() (*Limiter* method), 177
 add_listener() (*EventListenerGroup* method), 183
 add_mode() (*MultiModeItem* method), 189
 add_no_change_watcher() (*EventListenerGroup* method), 184
 add_no_update_watcher() (*EventListenerGroup* method), 183
 add_value() (*Statistics* method), 179
 aggregation_func() (*AggregationItem* method), 61
 aggregation_period() (*AggregationItem* method), 62
 aggregation_source() (*AggregationItem* method), 62
 AggregationItem (class in *HABApp.core.items*), 61
 all_modes() (*MultiModeItem* method), 189
 allow() (*Limiter* method), 178
 AndFilterGroup (class in *HABApp.core.events*), 35
 at() (*HABAppSchedulerView* method), 36

B

b (*HSB* property), 54
 b (*RGB* property), 52
 BaseValueItem (class in *HABApp.core.items*), 65
 blue (*RGB* property), 52
 boundary_func() (*DawnJob* method), 208
 boundary_func() (*DayOfWeekJob* method), 206
 boundary_func() (*DuskJob* method), 212
 boundary_func() (*ReoccurringJob* method), 205
 boundary_func() (*SunriseJob* method), 209
 boundary_func() (*SunsetJob* method), 211
 brightness (*HSB* property), 54
 buffer (*Connection* attribute), 19

C

ca_cert (*TLSSettings* attribute), 18
 calculate_value() (*MultiModeItem* method), 189
 CallItem (class in *HABApp.openhab.items*), 122
 cancel() (*CountdownJob* method), 204
 cancel() (*DawnJob* method), 208
 cancel() (*DayOfWeekJob* method), 207

cancel() (*DuskJob* method), 212
 cancel() (*EventListenerGroup* method), 183
 cancel() (*ItemNoChangeWatch* method), 203
 cancel() (*ItemNoUpdateWatch* method), 203
 cancel() (*OneTimeJob* method), 204
 cancel() (*ReoccurringJob* method), 205
 cancel() (*SunriseJob* method), 210
 cancel() (*SunsetJob* method), 211
 cancel() (*SwitchItemValueMode* method), 191
 cancel() (*ValueMode* method), 190
 ChannelTriggeredEvent (class in *HABApp.openhab.events*), 135
 closed() (*ContactItem* method), 76
 ColorItem (class in *HABApp.core.items*), 57
 ColorItem (class in *HABApp.openhab.items*), 97
 CompressedMidnightRotatingFileHandler (class in *HABApp.config.logging*), 29
 config (*DirectoriesConfig* attribute), 17
 connection (*MqttConfig* attribute), 18
 connection (*OpenhabConfig* attribute), 19
 ContactItem (class in *HABApp.openhab.items*), 76
 countdown() (*CountdownJob* method), 204
 countdown() (*HABAppSchedulerView* method), 37
 CountdownJob (class in *HABApp.rule.scheduler.jobs*), 204
 create_item() (in module *HABApp.openhab.interface_sync*), 129
 create_link() (in module *HABApp.openhab.interface_sync*), 131

D

DatetimeItem (class in *HABApp.openhab.items*), 89
 DawnJob (class in *HABApp.rule.scheduler.jobs*), 208
 DayOfWeekJob (class in *HABApp.rule.scheduler.jobs*), 206
 deactivate_listener() (*EventListenerGroup* method), 183
 delete() (in module *HABApp.rule.interfaces.http*), 169
 DictParameter (class in *HABApp.parameters*), 49
 DimmerItem (class in *HABApp.openhab.items*), 85
 directories (*ApplicationConfig* attribute), 17
 down() (*RollershutterItem* method), 93

DuskJob (class in HABApp.rule.scheduler.jobs), 212

E

earliest() (DawnJob method), 208
earliest() (DayOfWeekJob method), 207
earliest() (DuskJob method), 212
earliest() (ReoccurringJob method), 205
earliest() (SunriseJob method), 210
earliest() (SunsetJob method), 211
elevation (LocationConfig attribute), 17
enabled (Ping attribute), 20
enabled (SwitchItemValueMode property), 191
enabled (ThreadPoolConfig attribute), 21
enabled (TLSSettings attribute), 18
enabled (ValueMode property), 190
ensure_folder() (DirectoriesConfig class method), 17
EVENT (ItemNoChangeWatch attribute), 203
EVENT (ItemNoUpdateWatch attribute), 203
EventFilter (class in HABApp.core.events), 35
EventListenerGroup (class in HABApp.util), 183
every() (HABAppSchedulerView method), 37
every_hour() (HABAppSchedulerView method), 40
every_minute() (HABAppSchedulerView method), 39
execute_python() (Rule method), 44
execute_subprocess() (Rule method), 43

F

Fade (class in HABApp.util), 181
FinishedProcessInfo (class in HABApp.rule), 41
FixedWindowElasticExpiryLimitInfo (class in HABApp.util.rate_limiter.limiter), 178
flush_every (LoggingConfig attribute), 21
from_hsb() (RGB class method), 51
from_rgb() (HSB class method), 53

G

g (RGB property), 52
general (MqttConfig attribute), 18
general (OpenhabConfig attribute), 19
get() (in module HABApp.rule.interfaces.http), 169
get_client_session() (in module HABApp.rule.interfaces.http), 170
get_create_item() (AggregationItem class method), 62
get_create_item() (ColorItem class method), 57
get_create_item() (Item class method), 54
get_create_item() (MqttItem class method), 154
get_create_item() (MqttPairItem class method), 158
get_create_item() (MultiModeItem class method), 188
get_item() (AggregationItem class method), 62
get_item() (BaseValueItem class method), 65
get_item() (CallItem class method), 123
get_item() (ColorItem class method), 57, 98

get_item() (ContactItem class method), 76
get_item() (DatetimeItem class method), 89
get_item() (DimmerItem class method), 85
get_item() (GroupItem class method), 114
get_item() (ImageItem class method), 118
get_item() (in module HABApp.openhab.interface_sync), 128
get_item() (Item class method), 55
get_item() (LocationItem class method), 107
get_item() (MqttItem class method), 155
get_item() (MqttPairItem class method), 158
get_item() (NumberItem class method), 72
get_item() (PlayerItem class method), 111
get_item() (RollershutterItem class method), 93
get_item() (StringItem class method), 103
get_item() (SwitchItem class method), 81
get_item() (Thing class method), 127
get_items() (Rule static method), 44
get_link() (in module HABApp.openhab.interface_sync), 130
get_mode() (MultiModeItem method), 189
get_next_run() (CountdownJob method), 204
get_next_run() (DawnJob method), 208
get_next_run() (DayOfWeekJob method), 207
get_next_run() (DuskJob method), 213
get_next_run() (OneTimeJob method), 204
get_next_run() (ReoccurringJob method), 205
get_next_run() (SunriseJob method), 210
get_next_run() (SunsetJob method), 211
get_persistence_data() (CallItem method), 123
get_persistence_data() (ColorItem method), 98
get_persistence_data() (ContactItem method), 76
get_persistence_data() (DatetimeItem method), 89
get_persistence_data() (DimmerItem method), 85
get_persistence_data() (GroupItem method), 115
get_persistence_data() (ImageItem method), 119
get_persistence_data() (in module HABApp.openhab.interface_sync), 130
get_persistence_data() (LocationItem method), 107
get_persistence_data() (NumberItem method), 72
get_persistence_data() (PlayerItem method), 111
get_persistence_data() (RollershutterItem method), 93
get_persistence_data() (StringItem method), 103
get_persistence_data() (SwitchItem method), 81
get_persistence_services() (in module HABApp.openhab.interface_sync), 130
get_rgb() (ColorItem method), 57, 98
get_thing() (in module HABApp.openhab.interface_sync), 128
get_value() (AggregationItem method), 62
get_value() (BaseValueItem method), 65
get_value() (CallItem method), 123
get_value() (ColorItem method), 58, 98

[get_value\(\)](#) (*ContactItem* method), 77
[get_value\(\)](#) (*DatetimeItem* method), 90
[get_value\(\)](#) (*DimmerItem* method), 85
[get_value\(\)](#) (*Fade* method), 181
[get_value\(\)](#) (*GroupItem* method), 115
[get_value\(\)](#) (*ImageItem* method), 119
[get_value\(\)](#) (*Item* method), 55
[get_value\(\)](#) (*LocationItem* method), 107
[get_value\(\)](#) (*MqttItem* method), 155
[get_value\(\)](#) (*MqttPairItem* method), 158
[get_value\(\)](#) (*NumberItem* method), 73
[get_value\(\)](#) (*PlayerItem* method), 111
[get_value\(\)](#) (*RollershutterItem* method), 94
[get_value\(\)](#) (*StringItem* method), 103
[get_value\(\)](#) (*SwitchItem* method), 81
[green](#) (*RGB* property), 52
[GroupItem](#) (class in *HABApp.openhab.items*), 114
[GroupStateChangedEvent](#) (class in *HABApp.openhab.events*), 134

H

[h](#) (*HSB* property), 54
[habapp](#) (*ApplicationConfig* attribute), 17
[HABApp.openhab.interface_sync](#) (module), 128
[HABApp.rule.interfaces.http](#) (module), 169
[HABApp.util](#) (module), 171
[HABAppException](#) (class in *HABApp.core.events.habapp_events*), 163
[HABAppSchedulerView](#) (class in *HABApp.rule.scheduler*), 36
[hits](#) (*FixedWindowElasticExpiryLimitInfo* attribute), 178
[hits](#) (*LeakyBucketLimitInfo* attribute), 179
[host](#) (*Connection* attribute), 18
[HSB](#) (class in *HABApp.core.types*), 53
[hsb_to_rgb\(\)](#) (in module *HABApp.util.functions*), 174
[hue](#) (*HSB* property), 54

I

[identifier](#) (*Connection* attribute), 18
[ImageItem](#) (class in *HABApp.openhab.items*), 118
[info\(\)](#) (*Limitter* method), 178
[insecure](#) (*TLSSettings* attribute), 18
[interval](#) (*Ping* attribute), 20
[interval\(\)](#) (*ReoccurringJob* method), 205
[is_closed\(\)](#) (*ContactItem* method), 77
[is_down\(\)](#) (*RollershutterItem* method), 94
[is_finished](#) (*Fade* property), 181
[is_off\(\)](#) (*ColorItem* method), 58, 98
[is_off\(\)](#) (*DimmerItem* method), 85
[is_off\(\)](#) (*SwitchItem* method), 81

[is_on\(\)](#) (*ColorItem* method), 58, 98
[is_on\(\)](#) (*DimmerItem* method), 86
[is_on\(\)](#) (*SwitchItem* method), 81
[is_open\(\)](#) (*ContactItem* method), 77
[is_up\(\)](#) (*RollershutterItem* method), 94
[Item](#) (class in *HABApp.core.items*), 54
[item](#) (*Ping* attribute), 20
[item_exists\(\)](#) (in module *HABApp.openhab.interface_sync*), 128
[ItemAddedEvent](#) (class in *HABApp.openhab.events*), 133
[ItemCommandEvent](#) (class in *HABApp.openhab.events*), 132
[ItemCommandEventFilter](#) (class in *HABApp.openhab.events*), 138
[ItemNoChangeEvent](#) (class in *HABApp.core.events*), 69
[ItemNoChangeWatch](#) (class in *HABApp.core.items.base_item_watch*), 203
[ItemNoUpdateEvent](#) (class in *HABApp.core.events*), 68
[ItemNoUpdateWatch](#) (class in *HABApp.core.items.base_item_watch*), 203
[ItemRemovedEvent](#) (class in *HABApp.openhab.events*), 134
[ItemStateChangedEvent](#) (class in *HABApp.openhab.events*), 132
[ItemStateChangedEventFilter](#) (class in *HABApp.openhab.events*), 138
[ItemStateEvent](#) (class in *HABApp.openhab.events*), 132
[ItemStatePredictedEvent](#) (class in *HABApp.openhab.events*), 134
[ItemStateUpdatedEventFilter](#) (class in *HABApp.openhab.events*), 138
[ItemUpdatedEvent](#) (class in *HABApp.openhab.events*), 133

J

[jitter\(\)](#) (*DawnJob* method), 209
[jitter\(\)](#) (*DayOfWeekJob* method), 207
[jitter\(\)](#) (*DuskJob* method), 213
[jitter\(\)](#) (*ReoccurringJob* method), 205
[jitter\(\)](#) (*SunriseJob* method), 210
[jitter\(\)](#) (*SunsetJob* method), 211

L

[last_change](#) (*AggregationItem* property), 64
[last_change](#) (*BaseValueItem* property), 67
[last_change](#) (*CallItem* property), 126
[last_change](#) (*ColorItem* property), 60, 102
[last_change](#) (*ContactItem* property), 80
[last_change](#) (*DatetimeItem* property), 92
[last_change](#) (*DimmerItem* property), 89
[last_change](#) (*GroupItem* property), 118
[last_change](#) (*ImageItem* property), 122

last_change (*Item property*), 57
 last_change (*LocationItem property*), 110
 last_change (*MqttItem property*), 157
 last_change (*MqttPairItem property*), 160
 last_change (*NumberItem property*), 76
 last_change (*PlayerItem property*), 114
 last_change (*RollershutterItem property*), 97
 last_change (*StringItem property*), 106
 last_change (*SwitchItem property*), 84
 last_change (*Thing property*), 127
 last_update (*AggregationItem property*), 64
 last_update (*BaseValueItem property*), 67
 last_update (*CallItem property*), 126
 last_update (*ColorItem property*), 61, 102
 last_update (*ContactItem property*), 80
 last_update (*DatetimeItem property*), 92
 last_update (*DimmerItem property*), 89
 last_update (*GroupItem property*), 118
 last_update (*ImageItem property*), 122
 last_update (*Item property*), 57
 last_update (*LocationItem property*), 110
 last_update (*MqttItem property*), 157
 last_update (*MqttPairItem property*), 160
 last_update (*NumberItem property*), 76
 last_update (*PlayerItem property*), 114
 last_update (*RollershutterItem property*), 97
 last_update (*StringItem property*), 106
 last_update (*SwitchItem property*), 84
 last_update (*Thing property*), 128
 latest() (*DawnJob method*), 209
 latest() (*DayOfWeekJob method*), 207
 latest() (*DuskJob method*), 213
 latest() (*ReoccurringJob method*), 206
 latest() (*SunriseJob method*), 210
 latest() (*SunsetJob method*), 211
 latitude (*LocationConfig attribute*), 17
 LeakyBucketLimitInfo (class in *HABApp.util.rate_limiter.limiter*), 178
 lib (*DirectoriesConfig attribute*), 17
 limit (*FixedWindowElasticExpiryLimitInfo attribute*), 178
 limit (*LeakyBucketLimitInfo attribute*), 179
 Limiter (class in *HABApp.util.rate_limiter.limiter*), 177
 LimiterInfo (class in *HABApp.util.rate_limiter.limiter*), 178
 limits (*LimiterInfo attribute*), 178
 listen() (*EventListenerGroup method*), 183
 listen_event() (*AggregationItem method*), 62
 listen_event() (*BaseValueItem method*), 65
 listen_event() (*CallItem method*), 123
 listen_event() (*ColorItem method*), 58, 98
 listen_event() (*ContactItem method*), 77
 listen_event() (*DatetimeItem method*), 90
 listen_event() (*DimmerItem method*), 86

listen_event() (*GroupItem method*), 115
 listen_event() (*ImageItem method*), 119
 listen_event() (*Item method*), 55
 listen_event() (*ItemNoChangeWatch method*), 203
 listen_event() (*ItemNoUpdateWatch method*), 203
 listen_event() (*LocationItem method*), 107
 listen_event() (*MqttItem method*), 155
 listen_event() (*MqttPairItem method*), 158
 listen_event() (*NumberItem method*), 73
 listen_event() (*PlayerItem method*), 111
 listen_event() (*RollershutterItem method*), 94
 listen_event() (*Rule method*), 43
 listen_event() (*StringItem method*), 103
 listen_event() (*SwitchItem method*), 81
 listen_event() (*Thing method*), 127
 listen_only (*General attribute*), 19, 20
 location (*ApplicationConfig attribute*), 17
 LocationItem (class in *HABApp.openhab.items*), 106
 logging (*DirectoriesConfig attribute*), 17
 logging (*HABAppConfig attribute*), 20
 longitude (*LocationConfig attribute*), 17

M

max() (in module *HABApp.util.functions*), 173
 members (*GroupItem property*), 118
 MidnightRotatingFileHandler (class in *HABApp.config.logging*), 29
 min() (in module *HABApp.util.functions*), 173
 min_start_level (*General attribute*), 20
 min_uptime (*General attribute*), 20
 module
 HABApp.openhab.interface_sync, 128
 HABApp.rule.interfaces.http, 169
 HABApp.util, 171
 mqtt (*ApplicationConfig attribute*), 17
 mqtt (built-in class), 153
 MqttItem (class in *HABApp.mqtt.items*), 154
 MqttPairItem (class in *HABApp.mqtt.items*), 157
 MqttValueChangeEvent (class in *HABApp.mqtt.events*), 161
 MqttValueUpdateEvent (class in *HABApp.mqtt.events*), 161
 MultiModeItem (class in *HABApp.util.multimode*), 188

N

name (*AggregationItem property*), 64
 name (*BaseValueItem property*), 67
 name (*CallItem property*), 126
 name (*ColorItem property*), 61, 102
 name (*ContactItem property*), 80
 name (*DatetimeItem property*), 93
 name (*DimmerItem property*), 89
 name (*GroupItem property*), 118
 name (*ImageItem property*), 122

name (*Item* property), 57
 name (*LocationItem* property), 110
 name (*MqttItem* property), 157
 name (*MqttPairItem* property), 160
 name (*NumberItem* property), 76
 name (*PlayerItem* property), 114
 name (*RollershutterItem* property), 97
 name (*StringItem* property), 106
 name (*SwitchItem* property), 84
 name (*Thing* property), 128
 NoEventFilter (*class in HABApp.core.events*), 35
 NumberItem (*class in HABApp.openhab.items*), 72

O

off() (*ColorItem* method), 99
 off() (*DimmerItem* method), 86
 off() (*SwitchItem* method), 82
 offset() (*DawnJob* method), 209
 offset() (*DayOfWeekJob* method), 207
 offset() (*DuskJob* method), 213
 offset() (*ReoccurringJob* method), 206
 offset() (*SunriseJob* method), 210
 offset() (*SunsetJob* method), 212
 oh_post_update() (*CallItem* method), 123
 oh_post_update() (*ColorItem* method), 99
 oh_post_update() (*ContactItem* method), 77
 oh_post_update() (*DatetimeItem* method), 90
 oh_post_update() (*DimmerItem* method), 86
 oh_post_update() (*GroupItem* method), 115
 oh_post_update() (*ImageItem* method), 119
 oh_post_update() (*LocationItem* method), 107
 oh_post_update() (*NumberItem* method), 73
 oh_post_update() (*PlayerItem* method), 111
 oh_post_update() (*RollershutterItem* method), 94
 oh_post_update() (*StringItem* method), 103
 oh_post_update() (*SwitchItem* method), 82
 oh_post_update_if() (*CallItem* method), 123
 oh_post_update_if() (*ColorItem* method), 99
 oh_post_update_if() (*ContactItem* method), 77
 oh_post_update_if() (*DatetimeItem* method), 90
 oh_post_update_if() (*DimmerItem* method), 86
 oh_post_update_if() (*GroupItem* method), 115
 oh_post_update_if() (*ImageItem* method), 119
 oh_post_update_if() (*LocationItem* method), 107
 oh_post_update_if() (*NumberItem* method), 73
 oh_post_update_if() (*PlayerItem* method), 111
 oh_post_update_if() (*RollershutterItem* method), 94
 oh_post_update_if() (*StringItem* method), 104
 oh_post_update_if() (*SwitchItem* method), 82
 oh_send_command() (*CallItem* method), 124
 oh_send_command() (*ColorItem* method), 100
 oh_send_command() (*ContactItem* method), 78
 oh_send_command() (*DatetimeItem* method), 91
 oh_send_command() (*DimmerItem* method), 87
 oh_send_command() (*GroupItem* method), 116
 oh_send_command() (*ImageItem* method), 120
 oh_send_command() (*LocationItem* method), 108
 oh_send_command() (*NumberItem* method), 74
 oh_send_command() (*PlayerItem* method), 112
 oh_send_command() (*RollershutterItem* method), 95
 oh_send_command() (*StringItem* method), 104
 oh_send_command() (*SwitchItem* method), 82
 on() (*ColorItem* method), 100
 on() (*DimmerItem* method), 87
 on() (*SwitchItem* method), 83
 on_day_of_week() (*HABAppSchedulerView* method), 37
 on_every_day() (*HABAppSchedulerView* method), 38
 on_rule_loaded() (*Rule* method), 43
 on_rule_removed() (*Rule* method), 43
 on_sun_dawn() (*HABAppSchedulerView* method), 39
 on_sun_dusk() (*HABAppSchedulerView* method), 39
 on_sunrise() (*HABAppSchedulerView* method), 38
 on_sunset() (*HABAppSchedulerView* method), 38
 on_weekends() (*HABAppSchedulerView* method), 39
 on_workdays() (*HABAppSchedulerView* method), 40
 OneTimeJob (*class in HABApp.rule.scheduler.jobs*), 204
 open() (*ContactItem* method), 78
 openhab (*ApplicationConfig* attribute), 17
 OrFilterGroup (*class in HABApp.core.events*), 35

P

param (*DirectoriesConfig* attribute), 17
 Parameter (*class in HABApp.parameters*), 49
 parse_limits() (*Limiter* method), 177
 password (*Connection* attribute), 18, 19
 percent() (*ColorItem* method), 100
 percent() (*DimmerItem* method), 87
 percent() (*RollershutterItem* method), 95
 ping (*OpenhabConfig* attribute), 19
 PlayerItem (*class in HABApp.openhab.items*), 110
 port (*Connection* attribute), 18
 post() (*in module HABApp.rule.interfaces.http*), 170
 post_event() (*Rule* method), 43
 post_rgb() (*ColorItem* method), 58, 100
 post_update() (*in module HABApp.openhab.interface_sync*), 131
 post_value() (*AggregationItem* method), 63
 post_value() (*BaseValueItem* method), 65
 post_value() (*CallItem* method), 124
 post_value() (*ColorItem* method), 59, 100
 post_value() (*ContactItem* method), 78
 post_value() (*DatetimeItem* method), 91
 post_value() (*DimmerItem* method), 87
 post_value() (*GroupItem* method), 116
 post_value() (*ImageItem* method), 120
 post_value() (*Item* method), 55
 post_value() (*LocationItem* method), 108

`post_value()` (*MqttItem* method), 155
`post_value()` (*MqttPairItem* method), 158
`post_value()` (*NumberItem* method), 74
`post_value()` (*PlayerItem* method), 112
`post_value()` (*RollershutterItem* method), 95
`post_value()` (*StringItem* method), 104
`post_value()` (*SwitchItem* method), 83
`post_value_if()` (*AggregationItem* method), 63
`post_value_if()` (*BaseValueItem* method), 66
`post_value_if()` (*CallItem* method), 124
`post_value_if()` (*ColorItem* method), 59, 100
`post_value_if()` (*ContactItem* method), 79
`post_value_if()` (*DatetimeItem* method), 91
`post_value_if()` (*DimmerItem* method), 87
`post_value_if()` (*GroupItem* method), 116
`post_value_if()` (*ImageItem* method), 120
`post_value_if()` (*Item* method), 55
`post_value_if()` (*LocationItem* method), 108
`post_value_if()` (*MqttItem* method), 155
`post_value_if()` (*MqttPairItem* method), 159
`post_value_if()` (*NumberItem* method), 74
`post_value_if()` (*PlayerItem* method), 112
`post_value_if()` (*RollershutterItem* method), 95
`post_value_if()` (*StringItem* method), 105
`post_value_if()` (*SwitchItem* method), 83
`publish` (*MqttConfig* attribute), 18
`publish()` (*mqtt* method), 153
`publish()` (*MqttItem* method), 156
`publish()` (*MqttPairItem* method), 159
`put()` (in module *HABApp.rule.interfaces.http*), 170

Q

`qos` (*Publish* attribute), 19
`qos` (*Subscribe* attribute), 18

R

`r` (*RGB* property), 52
`RateLimiter()` (in module *HABApp.util*), 177
`red` (*RGB* property), 52
`remaining()` (*CountdownJob* method), 204
`remaining()` (*DawnJob* method), 209
`remaining()` (*DayOfWeekJob* method), 207
`remaining()` (*DuskJob* method), 213
`remaining()` (*OneTimeJob* method), 204
`remaining()` (*ReoccurringJob* method), 206
`remaining()` (*SunriseJob* method), 210
`remaining()` (*SunsetJob* method), 212
`remove_item()` (in module *HABApp.openhab.interface_sync*), 129
`remove_link()` (in module *HABApp.openhab.interface_sync*), 130
`remove_metadata()` (in module *HABApp.openhab.interface_sync*), 129
`remove_mode()` (*MultiModeItem* method), 188

`ReoccurringJob` (class in *HABApp.rule.scheduler.jobs*), 205
`replace()` (*HSB* method), 53
`replace()` (*RGB* method), 52
`RequestFileLoadEvent` (class in *HABApp.core.events.habapp_events*), 163
`RequestFileUnloadEvent` (class in *HABApp.core.events.habapp_events*), 163
`reset()` (*Limiter* method), 178
`retain` (*Publish* attribute), 19
`RGB` (class in *HABApp.core.types*), 51
`rgb_to_hsb()` (in module *HABApp.util.functions*), 174
`RollershutterItem` (class in *HABApp.openhab.items*), 93
`Rule` (class in *HABApp*), 43
`rules` (*DirectoriesConfig* attribute), 17

S

`s` (*HSB* property), 54
`saturation` (*HSB* property), 54
`schedule_fade()` (*Fade* method), 181
`send_command()` (in module *HABApp.openhab.interface_sync*), 131
`set_enabled()` (*Thing* method), 127
`set_enabled()` (*ValueMode* method), 190
`set_file_validator()` (in module *HABApp.parameters*), 48
`set_metadata()` (in module *HABApp.openhab.interface_sync*), 129
`set_persistence_data()` (in module *HABApp.openhab.interface_sync*), 130
`set_rgb()` (*ColorItem* method), 59, 101
`set_thing_enabled()` (in module *HABApp.openhab.interface_sync*), 128
`set_value()` (*AggregationItem* method), 63
`set_value()` (*BaseValueItem* method), 66
`set_value()` (*CallItem* method), 125
`set_value()` (*ColorItem* method), 60, 101
`set_value()` (*ContactItem* method), 79
`set_value()` (*DatetimeItem* method), 92
`set_value()` (*DimmerItem* method), 88
`set_value()` (*GroupItem* method), 117
`set_value()` (*ImageItem* method), 121
`set_value()` (*Item* method), 56
`set_value()` (*LocationItem* method), 109
`set_value()` (*MqttItem* method), 156
`set_value()` (*MqttPairItem* method), 160
`set_value()` (*NumberItem* method), 75
`set_value()` (*PlayerItem* method), 113
`set_value()` (*RollershutterItem* method), 96
`set_value()` (*StringItem* method), 105
`set_value()` (*SwitchItem* method), 84
`set_value()` (*SwitchItemValueMode* method), 191
`set_value()` (*ValueMode* method), 190

setup() (*Fade method*), 181
 skips (*FixedWindowElasticExpiryLimitInfo attribute*), 178
 skips (*LeakyBucketLimitInfo attribute*), 179
 skips (*LimiterInfo attribute*), 178
 soon() (*HABAppSchedulerView method*), 39
 Statistics (*class in HABApp.util*), 179
 stop() (*CountdownJob method*), 205
 stop_fade() (*Fade method*), 181
 StringItem (*class in HABApp.openhab.items*), 102
 subscribe (*MqttConfig attribute*), 18
 subscribe() (*mqtt method*), 153
 SunriseJob (*class in HABApp.rule.scheduler.jobs*), 209
 SunsetJob (*class in HABApp.rule.scheduler.jobs*), 211
 SwitchItem (*class in HABApp.openhab.items*), 80
 SwitchItemValueMode (*class in HABApp.util.multimode*), 191

T

test_allow() (*Limiter method*), 178
 Thing (*class in HABApp.openhab.items*), 126
 ThingAddedEvent (*class in HABApp.openhab.events*), 135
 ThingFirmwareStatusInfoEvent (*class in HABApp.openhab.events*), 137
 ThingRemovedEvent (*class in HABApp.openhab.events*), 136
 ThingStatusInfoChangedEvent (*class in HABApp.openhab.events*), 137
 ThingStatusInfoEvent (*class in HABApp.openhab.events*), 136
 ThingUpdatedEvent (*class in HABApp.openhab.events*), 136
 thread_pool (*HABAppConfig attribute*), 20
 threads (*ThreadPoolConfig attribute*), 21
 time() (*DayOfWeekJob method*), 207
 time_remaining (*FixedWindowElasticExpiryLimitInfo attribute*), 178
 time_remaining (*LeakyBucketLimitInfo attribute*), 179
 tls (*Connection attribute*), 18
 to_hsb() (*RGB method*), 52
 to_item() (*CountdownJob method*), 205
 to_item() (*DawnJob method*), 209
 to_item() (*DayOfWeekJob method*), 208
 to_item() (*DuskJob method*), 213
 to_item() (*OneTimeJob method*), 204
 to_item() (*ReoccurringJob method*), 206
 to_item() (*SunriseJob method*), 211
 to_item() (*SunsetJob method*), 212
 to_rgb() (*HSB method*), 53
 to_str() (*HABAppException method*), 163
 toggle() (*SwitchItem method*), 84
 topic_filter (*Connection attribute*), 19
 topics (*Subscribe attribute*), 18

total_skips (*Limiter property*), 177
 total_skips (*LimiterInfo attribute*), 178

U

unit (*NumberItem property*), 76
 unsubscribe() (*mqtt method*), 153
 up() (*RollershutterItem method*), 96
 update() (*Statistics method*), 179
 url (*Connection attribute*), 19
 use_buffer (*LoggingConfig attribute*), 21
 user (*Connection attribute*), 18, 19

V

value (*DictParameter property*), 49
 value (*Parameter property*), 49
 value (*SwitchItemValueMode property*), 191
 value (*ValueMode property*), 190
 ValueChangeEvent (*class in HABApp.core.events*), 68
 ValueChangeEventFilter (*class in HABApp.core.events*), 35
 ValueMode (*class in HABApp.util.multimode*), 190
 ValueUpdateEvent (*class in HABApp.core.events*), 68
 ValueUpdateEventFilter (*class in HABApp.core.events*), 35
 verify_ssl (*Connection attribute*), 19

W

wait_for_openhab (*General attribute*), 20
 watch_change() (*AggregationItem method*), 64
 watch_change() (*BaseValueItem method*), 67
 watch_change() (*CallItem method*), 125
 watch_change() (*ColorItem method*), 60, 101
 watch_change() (*ContactItem method*), 79
 watch_change() (*DatetimeItem method*), 92
 watch_change() (*DimmerItem method*), 88
 watch_change() (*GroupItem method*), 117
 watch_change() (*ImageItem method*), 121
 watch_change() (*Item method*), 56
 watch_change() (*LocationItem method*), 109
 watch_change() (*MqttItem method*), 156
 watch_change() (*MqttPairItem method*), 160
 watch_change() (*NumberItem method*), 75
 watch_change() (*PlayerItem method*), 113
 watch_change() (*RollershutterItem method*), 96
 watch_change() (*StringItem method*), 105
 watch_change() (*SwitchItem method*), 84
 watch_change() (*Thing method*), 127
 watch_update() (*AggregationItem method*), 64
 watch_update() (*BaseValueItem method*), 67
 watch_update() (*CallItem method*), 126
 watch_update() (*ColorItem method*), 60, 102
 watch_update() (*ContactItem method*), 80
 watch_update() (*DatetimeItem method*), 92
 watch_update() (*DimmerItem method*), 88

`watch_update()` (*GroupItem method*), 117
`watch_update()` (*ImageItem method*), 122
`watch_update()` (*Item method*), 56
`watch_update()` (*LocationItem method*), 110
`watch_update()` (*MqttItem method*), 157
`watch_update()` (*MqttPairItem method*), 160
`watch_update()` (*NumberItem method*), 75
`watch_update()` (*PlayerItem method*), 114
`watch_update()` (*RollershutterItem method*), 96
`watch_update()` (*StringItem method*), 106
`watch_update()` (*SwitchItem method*), 84
`watch_update()` (*Thing method*), 127
`weekdays()` (*DayOfWeekJob method*), 208