HABApp Documentation

Release beta

spacemanspiff2007

Nov 12, 2019

USER DOCUMENTATION

CHAPTER

ONE

INSTALLATION & USAGE

1.1 Virtual environment

1.1.1 Installation

Hint: HABApp requires at least Python 3.6

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

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

   ```
   cd /opt
   ```

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

   ```
   python3 -m venv habapp
   ```

3. Go into folder of virtual environment:

   ```
   cd habapp
   ```

4. Activate the virtual environment

   Linux:

   ```
   source bin/activate
   ```

   Windows:

   ```
   Scripts\activate
   ```

5. Upgrade pip:

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

6. Install HABApp:

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

7. Run HABApp:

```
habapp --config PATH_TO_CONFIGURATION_FOLDER
```

1.1.2 Upgrading

1. Stop HABApp
2. Run the following command:

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

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

1.1.3 Error message while installing ujson

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

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

1.1.4 Error message while installing ruamel.yaml

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

Run the follwing command to fix it:

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

1.1.5 Autostart after reboot

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:

```
[Unit]
Description=HABApp
After=network-online.target

[Service]
Type=simple
User=openhab
Group=openhab
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
```

1.2 Docker

1.2.1 Installation

Installation through docker is also available:

```
docker pull spacemanspiff2007/habapp
```

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

1.2.2 Updating docker on Synology

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

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

1.3 Upgrading to a newer version

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

1.4 HABApp arguments

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

```
habapp -h
```

```
usage: -c [-h] [-c CONFIG] [-s SLEEP] [--NoMQTTConnectionErrors]

Start HABApp

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Path to configuration folder (where the config.yml is
                        located)
  -s SLEEP, --sleep SLEEP
                        Sleep time in seconds before starting HABApp
  --NoMQTTConnectionErrors
                        Suppress MQTT connection errors and only log them (for
                        testing without a connected MQTT Broker)
```

CHAPTER

TWO

CONFIGURATION

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

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

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

2.1 Configuration contents

```
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
    lib: lib        # Custom modules, libraries and files can be placed there
    param: param    # Optional, this is the folder where the parameter files will be␣
↪created and loaded from

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 item
        interval: 10          # Seconds between two pings

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

CHAPTER THREE

LOGGING

3.1 Example usage

The logging library is the standard python library.

```
import logging

import HABApp

log = logging.getLogger('MyRule')


class MyLoggingRule(HABApp.Rule):

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

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


MyLoggingRule()
```

3.2 Example configuration

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

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

Hint:

Is is recommended to use absolute paths as filenames
e.g.: `/HABApp/logs/my_logfile.log` or `c:\HABApp\logs\my_logfile.log` on Windows

```
# required, can not be omitted but does nothing
version : 1

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

# describes the available file handlers
handlers:
  HABApp_default:
    class: logging.handlers.RotatingFileHandler
    filename: 'HABApp.log'
    maxBytes: 10_000_000
    backupCount: 3

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

  MyRuleHandler:
    class: logging.handlers.RotatingFileHandler
    filename: 'c:\HABApp\Logs\MyRule.log'    # absolute filename is recommended
    maxBytes: 10_000_000
    backupCount: 3

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


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

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

CHAPTER FOUR

RULE

4.1 Interacting with items

Items are like variables. They have a name and a state (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.

Function	Description
item_watch()	Keep watch on the state of an item. If the item does not receive an update or change for a certain amount of time there will be a `ValueNoUpdateEvent` or a `ValueNoChangeEvent` on the event bus. It is possible to add multiple watches to an item.
item_watch_and_listen()	Convenience function which combines *item_watch* and *listen_event*

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:
```

(continues on next page)

(continued from previous page)

```
    pass    # do something

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

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

Simple item

Variables

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

Returns the item

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

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

Parameters

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

Returns item

set_value(*new_value*)

Set a new value without creating events on the event bus

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

Return type bool

Returns True if state has changed

post_value(*new_value*)

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

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

get_value(*default_value=None*)

Return the value of the item.

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

Return type Any

Returns value of the item

4.2 Events

Function	Description
listen_event()	Add a function which will be called as soon as an event occurs. The event will be passed as an argument into the function. There is the possibility to specify the event class which will reduce the function calls accordingly (typically `ValueUpdateEvent` or `ValueChangeEvent`).

Example:

```
def __init__(self):
    super().__init__()
    self.listen_event('MyOpenhabItem', self.on_change, ValueChangeEvent)
    self.listen_event('My/MQTT/Topic', self.on_update, ValueUpdateEvent)

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

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

4.3 Scheduler

With the scheduler it is easy to call functions in the future or periodically. Do not use *time.sleep* but rather *run_in()*.

Function	Description
run_soon()	Run the callback as soon as possible (typically in the next second).
run_in()	Run the callback in x seconds.
run_at()	Run a function at a specified date_time
run_every()	Run a function periodically
run_minutely()	Run a function every minute
run_hourly()	Run a function every hour
run_daily()	Run a function every day
run_on_every_day()	Run a function at a specific time every day
run_on_workdays()	Run a function at a specific time on workdays
run_on_weekends()	Run a function at a specific time on weekends
run_on_day_of_week()	Run a function at a specific time on specific days of the week

All functions return an instance of ScheduledCallback

class `HABApp.rule.scheduler.`**`ScheduledCallback`**(*date_time*, *callback*, **args*, ***kwargs*)

`get_next_call`()
  Return the next execution timestamp

`check_due`(*now*)
  Check whether the callback is due for execution

  Parameters **`now`** (`datetime`) –

  Returns

execute()

Try to execute callback. If the callback is not due yet or execution has already finished nothing will happen

Return type `bool`

Returns True if callback has been executed else False

cancel()

Cancel execution

4.4 Running external tools

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

```
import HABApp

class MyExecutionRule(HABApp.Rule):

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

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

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

MyExecutionRule()
```

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

Information about the finished process.

Variables

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

4.5 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 this method provides syntax checks and auto complete.

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.

Example:

```
if typing.TYPE_CHECKING:
    from .class_b import ClassB
```

(continues on next page)

(continued from previous page)

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

    def function_a(self):
        # Important: always look up rule every time, never assign to a class member!
        r = self.get_rule('NameOfRuleB')  # type: ClassB
        r.function_b()
```

4.6 All available functions

class HABApp.**Rule**

Variables

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

item_watch(*name*, *seconds_constant*, *watch_only_changes=True*)

Keep watch on the state of an item.

if *watch_only_changes* is True (default) and the state does not change for *seconds_constant* a *ValueNoChangeEvent* will be sent to the event bus.

if *watch_only_changes* is False and the state does not receive and update for *seconds_constant* a *ValueNoUpdateEvent* will be sent to the event bus.

Parameters

- **name** (`Union[str, Item]`) – item name or item that shall be watched
- **seconds_constant** (`int`) – the amount of seconds the item has to be constant or has not received an update
- **watch_only_changes** –

Return type `WatchedItem`

item_watch_and_listen(*name*, *seconds_constant*, *callback*, *watch_only_changes=True*)

Convenience function which combines `item_watch` and `listen_event`

Parameters

- **name** (`str`) – item name
- **seconds_constant** (`int`) –
- **callback** – callback that accepts one parameter which will contain the event
- **watch_only_changes** –

Return type `Tuple[WatchedItem, EventBusListener]`

Returns

post_event (*name*, *event*)

Post an event to the event bus

Parameters

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

Returns

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

Register an event listener

Parameters

- **name** (`Union[Item, str, None]`) – item or name to listen to. Use None to listen to all events
- **callback** – callback that accepts one parameter which will contain the event
- **even_type** (`Union[AllEvents, Any]`) – Event filter. This is typically `ValueUpdateEvent` or `ValueChangeEvent` which will also trigger on changes/update from openhab or mqtt.

Return type `EventBusListener`

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

Run another program

Parameters

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

Returns

run_every (*time*, *interval*, *callback*, **args*, ***kwargs*)

Run a function periodically

Parameters

- **time** (`Union[None, datetime, timedelta, time]`) – Use a datetime.time object to specify a certain time of day, a datetime.timedelta object to specify a time in the future or None to use the current time.
- **interval** –
- **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 `ReoccurringScheduledCallback`

run_on_day_of_week (*time*, *weekdays*, *callback*, **args*, ***kwargs*)

Parameters

- **`time`** (`Union[None, datetime, timedelta, time]`) – Use a datetime.time object to specify a certain time of day, a datetime.timedelta object to specify a time in the future or None to use the current time.
- **`weekdays`** –
- **`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 `DayOfWeekScheduledCallback`

`run_on_every_day`(*time*, *callback*, **args*, ***kwargs*)

Parameters

- **`time`** (`Union[None, datetime, timedelta, time]`) – Use a datetime.time object to specify a certain time of day, a datetime.timedelta object to specify a time in the future or None to use the current time.
- **`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 `DayOfWeekScheduledCallback`

`run_on_workdays`(*time*, *callback*, **args*, ***kwargs*)

Parameters

- **`time`** (`Union[None, datetime, timedelta, time]`) – Use a datetime.time object to specify a certain time of day, a datetime.timedelta object to specify a time in the future or None to use the current time.
- **`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 `WorkdayScheduledCallback`

`run_on_weekends`(*time*, *callback*, **args*, ***kwargs*)

Parameters

- **`time`** (`Union[None, datetime, timedelta, time]`) – Use a datetime.time object to specify a certain time of day, a datetime.timedelta object to specify a time in the future or None to use the current time.
- **`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 `WeekendScheduledCallback`

`run_daily`(*callback*, **args*, ***kwargs*)

Picks a random hour, minute and second and runs the callback every day

Parameters

- **`callback`** – Function which will be called

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

Return type `ReoccurringScheduledCallback`

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

Parameters

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

Return type `ReoccurringScheduledCallback`

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

Parameters

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

Return type `ReoccurringScheduledCallback`

`run_at`(*date_time*, *callback*, **args*, ***kwargs*)
Run a function at a specified date_time

Parameters

- **`date_time`** (`Union`[`None`, `datetime`, `timedelta`, `time`]) –
- **`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 `ScheduledCallback`

`run_in`(*seconds*, *callback*, **args*, ***kwargs*)
Run the callback in x seconds

Parameters

- **`seconds`** (*`int`*) – Wait time in seconds or a timedelta obj before calling the function
- **`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 `ScheduledCallback`

`run_soon`(*callback*, **args*, ***kwargs*)
Run the callback as soon as possible (typically in the next second).

Parameters

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

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

Return type `ScheduledCallback`

`register_on_unload`(*func*)

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

Parameters **`func`** – function which will be called

CHAPTER

FIVE

PARAMETERS

5.1 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).

```
import HABApp

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

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

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

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

    def on_change_event( 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`.

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

`__init__`(*filename*, **keys*, *default_value='ToDo'*)
Class to dynamically access parameters which are loaded from file.

Parameters

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

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

Return type `Any`

5.2 Validation

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

`HABApp.parameters.`**`set_file_validator`**(*filename*, *validator*, *allow_extra_keys=True*)
Add a validator for the parameter file. If the file is already loaded this will reload the file.

Parameters

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

Example

```
import HABApp
import voluptuous

# Validator can even and should be specified before loading rules

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

```

(continues on next page)

(continued from previous page)

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

CHAPTER

SIX

OPENHAB

6.1 Interaction with a openHAB

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

6.2 Function parameters

class `HABApp.openhab.oh_interface.`**`OpenhabInterface`**(*connection*)

`post_update`(*item_name*, *state*)

Post an update to the item

Parameters

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

`send_command`(*item_name*, *command*)

Send the specified command to the item

Parameters

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

`create_item`(*item_type*, *name*, *label=”*, *category=”*, *tags=[]*, *groups=[]*, *group_type=”*, *group_function=”*, *group_function_params=[]*)

Creates a new item in the OpenHAB item registry or updates an existing one

Parameters

- **`item_type`** (`str`) – item type
- **`name`** (`str`) – item name
- **`label`** – item label
- **`category`** – item category
- **`tags`** (`List[str]`) – item tags
- **`groups`** (`List[str]`) – in which groups is the item
- **`group_type`** (`str`) – what kind of group is it

- **group_function** (`str`) – group state aggregation function
- **group_function_params** (`List[str]`) – params for group state aggregation

Returns True if item was created/updated

get_item(*item_name*)
Return the complete OpenHAB item definition

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

Return type *OpenhabItemDefinition*

remove_item(*item_name*)
Removes an item from the openHAB item registry

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

item_exists(*item_name*)
Check if an item exists in the OpenHAB item registry

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

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

Parameters

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

Returns

remove_metadata(*item_name*, *namespace*)
Remove metadata from an item

Parameters

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

Returns

class HABApp.openhab.oh_interface.**OpenhabItemDefinition**(*type*, *name*, *state*, *label=”*, *category=”*, *editable=True*, *tags=<factory>*, *groups=<factory>*, *members=<factory>*)

Variables

- **type** (*str*) – item type
- **name** (*str*) – item name
- **state** (*str*) – item state
- **state** – item label
- **category** (*str*) – item category
- **editable** (*bool*) – item can changed through Rest API

- **tags** (*typing.List[str]*) – item tags
- **groups** (*typing.List[str]*) – groups the item is in
- **members** (*typing.List[OpenhabItemDefinition]*) – If the item is a group this contains the members

6.3 Openhab item types

Openhab items are mapped to special classes and provide convenience functions.

Example:

```
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!
```

Openhab type	HABApp class
Contact	*ContactItem*
Switch	*SwitchItem*
Dimmer	*DimmerItem*
Rollershutter	*RollershutterItem*
Color	*ColorItem*

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

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

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

Returns item

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

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

Returns the item

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

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

Return type Any

Returns value of the item

is_closed()
Test value against closed-value

Return type `bool`

is_open()
Test value against open-value

Return type `bool`

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

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

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

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

Returns item

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

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

Returns the item

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

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

Return type `Any`

Returns value of the item

is_off()
Test value against off-value

Return type `bool`

is_on()
Test value against on-value

Return type `bool`

off()
Command item off

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

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

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

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

Returns item

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

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

Returns the item

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

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

Return type `Any`

Returns value of the item

is_off()
Test value against off-value

Return type `bool`

is_on()
Test value against on-value

Return type `bool`

off()
Command item off

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

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

down()
Command down

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

- **name** (`str`) – item name

- **`initial_value`** – state the item will have if it gets created

Returns item

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

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

Returns the item

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

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

Return type `Any`

Returns value of the item

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

up()
Command up

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

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

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

Returns item

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

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

Returns the item

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

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

Return type `Tuple[int, int, int]`

Returns rgb tuple

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

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

Return type `Any`

Returns value of the item

is_off()
Return true if item is off

Return type `bool`

is_on()
Return true if item is on

Return type `bool`

off()
Command item off

on()
Command item on

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

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

Parameters

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

set_rgb(*r*, *g*, *b*, *max_rgb_value=255*)
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

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 %)

6.4 Example openHAB rule

```python
import HABApp
from HABApp.core.events import ValueUpdateEvent, ValueChangeEvent
from HABApp.openhab.events import ItemStateEvent, ItemCommandEvent,␣
↪ItemStateChangedEvent
from HABApp.openhab.items import SwitchItem

class MyOpenhabRule(HABApp.Rule):

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

        # Trigger on item updates
        self.listen_event( 'TestContact', self.item_state_update, ItemStateEvent)
        self.listen_event( 'TestDateTime', self.item_state_update, ValueUpdateEvent)

        # Trigger on item changes
        self.listen_event( 'TestDateTime', self.item_state_change,␣
↪ItemStateChangedEvent)
        self.listen_event( 'TestSwitch', self.item_state_change, ValueChangeEvent)

        # Trigger on item commands
        self.listen_event( 'TestSwitch', 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_create_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()
```

CHAPTER

SEVEN

MQTT

7.1 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.

7.2 Rule Interface

`class mqtt`

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

Parameters

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

Returns 0 if successful

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

Parameters

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

Returns 0 if successful

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

Parameters **`topic`** – MQTT topic

Returns 0 if successful

7.3 MqttItem

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

```python
from HABApp.mqtt.items import MqttItem

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

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

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

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

classmethod get_create_item(*name*, *initial_value=None*)
Creates a new item in HABApp and returns it or returns the already existing one with the given name

Parameters

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

Returns item

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

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

Returns the item

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

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

Return type `Any`

Returns value of the item

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

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

Parameters

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

- **retain** (Optional[bool]) – retain message. If not specified value from configuration file will be used.

Returns 0 if successful

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

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

Return type bool

Returns True if state has changed

7.4 Example MQTT rule

```python
import datetime
import random

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

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

        self.run_every(
            time=datetime.timedelta(seconds=10),
            interval=datetime.timedelta(seconds=20),
            callback=self.publish_rand_value
        )

        self.my_mqtt_item = MqttItem.get_create_item('test/test')

        self.listen_event('test/test', self.topic_updated, ValueUpdateEvent)

    def publish_rand_value(self):
        print('test mqtt_publish')
        self.my_mqtt_item.publish(str(random.randint(0, 1000)))

    def topic_updated(self, event):
        assert isinstance(event, ValueUpdateEvent), type(event)
        print( f'mqtt topic "test/test" updated to {event.value}')


ExampleMqttTestRule()
```

CHAPTER

EIGHT

ADVANCED USAGE

8.1 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 rules when a parameter file gets reloaded (e.g. when `Parameter` is used to create rules dynamically) or an own notifier in case there are errors (e.g. Pushover).

Topic	Description	Events
HABApp.Rules	The corresponding events trigger a load/unload of the rule file specified in the event	`RequestFileLoadEvent` and `RequestFileUnloadEvent`
HABApp.Parameters	The corresponding events trigger a load/unload of the parameter file specified in the event	`RequestFileLoadEvent` and `RequestFileUnloadEvent`
HABApp.Errors	All errors in functions and rules of HABApp create an according event. Use this topic to create an own notifier in case of errors (e.g. Pushover).	`HABAppError`

class `HABApp.core.events.habapp_events.`**`RequestFileLoadEvent`**(*name*)

Request (re-) loading of the specified file

Variables **`filename`** (`str`) – relative filename

class `HABApp.core.events.habapp_events.`**`RequestFileUnloadEvent`**(*name*)

Request unloading of the specified file

Variables **`filename`** (`str`) – relative filename

class `HABApp.core.events.habapp_events.`**`HABAppError`**(*func_name*, *exception*, *traceback*)

Contains information about an error in a function

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`

CHAPTER NINE

ASYNCIO

Warning:

Please make sure you know what you are doing when using async functions!

If you have no asyncio experience please do not use this! The use of blocking calls in async functions will prevent HABApp from working properly!

9.1 async http

Async http calls are available through the `self.async_http` object in rule instances.

9.1.1 Functions

class `HABApp.rule.interfaces.`**`AsyncHttpConnection`**

`get` (*url*, *params=None*, ***kwargs*)
http get request

Parameters

- **`url`** (`str`) – Request URL
- **`params`** (`Optional[Mapping[str, str]]`) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **`data`** – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **`json`** – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **`kwargs`** (`Any`) – See aiohttp request for further possible kwargs

Return type `_RequestContextManager`

Returns awaitable

`post` (*url*, *params=None*, *data=None*, *json=None*, ***kwargs*)
http post request

Parameters

- **`url`** (`str`) – Request URL

- **params** (Optional[Mapping[str, str]]) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **data** (Optional[Any]) – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **json** (Optional[Any]) – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **kwargs** (Any) – See aiohttp request for further possible kwargs

Return type _RequestContextManager

Returns awaitable

put (*url*, *params=None*, *data=None*, *json=None*, ***kwargs*)
http put request

Parameters

- **url** (str) – Request URL
- **params** (Optional[Mapping[str, str]]) – Mapping, iterable of tuple of key/value pairs (e.g. dict) to be sent as parameters in the query string of the new request. Params example
- **data** (Optional[Any]) – Dictionary, bytes, or file-like object to send in the body of the request (optional)
- **json** (Optional[Any]) – Any json compatible python object, json and data parameters could not be used at the same time. (optional)
- **kwargs** (Any) – See aiohttp request for further possible kwargs

Return type _RequestContextManager

Returns awaitable

get_client_session ()
Return the aiohttp client session object for use in aiohttp libraries

Return type ClientSession

Returns session object

9.1.2 Examples

```
import asyncio

import HABApp


class AsyncRule(HABApp.Rule):

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

        self.run_soon(self.async_func)

    async def async_func(self):
        await asyncio.sleep(2)
```

(continues on next page)

(continued from previous page)

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


AsyncRule()
```

CHAPTER

TEN

UTIL - HELPERS AND UTILITIES

The util package contains useful classes which make rule creation easier.

10.1 CounterItem

10.1.1 Example

```
from HABApp.util import CounterItem
c = CounterItem.get_create_item('MyCounter', initial_value=5)
print(c.increase())
print(c.decrease())
print(c.reset())
```

```
6
5
5
```

10.1.2 Documentation

class HABApp.util.**CounterItem**(*name*, *initial_value=0*)

Implements a simple thread safe counter

__init__(*name*, *initial_value=0*)

Parameters **initial_value** (`int`) – Initial value of the counter

reset()

Reset value to initial value

increase(*step=1*)

Increase value

Parameters **step** – increase by this value, default = 1

Return type `int`

Returns value of the counter

decrease(*step=1*)

Decrease value

Parameters **step** – decrease by this value, default = 1

Return type `int`

Returns value of the counter

10.2 Statistics

10.2.1 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>
```

10.2.2 Documentation

class `HABApp.util.`**`Statistics`**(*max_age=None*, *max_samples=None*)

Calculate mathematical statistics of numerical values.

Variables

- **`sum`** – sum of all values
- **`min`** – minimum of all values
- **`max`** – maximum of all values
- **`mean`** – mean of all values
- **`median`** – median of all values
- **`last_value`** – last added value
- **`last_change`** – timestamp the last time a value was added

`__init__`(*max_age=None*, *max_samples=None*)

Parameters

- **`max_age`** – Maximum age of values in seconds
- **`max_samples`** – Maximum amount of samples which will be kept

`update`()

update values without adding a new value

`add_value`(*value*)

Add a new value and recalculate statistical values

Parameters **`value`** – new value

10.3 MultiModeItem

Prioritizer item which automatically switches between values with different priorities. Very useful when different states or modes overlap, e.g. automatic and manual mode. etc.

10.3.1 Basic Example

```
import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.util import MultiModeItem

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

        # create a new MultiModeItem
        item = MultiModeItem.get_create_item('MultiModeTestItem')
        self.listen_event(item, self.item_update, ValueUpdateEvent)

        # create two different modes which we will use
        item.create_mode('Automatic', 0, initial_value=5)
        item.create_mode('Manual', 10, initial_value=0)

        # This shows how to enable/disable a mode
        print('disable/enable the higher priority mode')
        item.get_mode('manual').set_enabled(False)
        item.get_mode('manual').set_value(11)

        # This shows that changes of the lower priority only show when
        # the mode with the higher priority gets disabled
        print('')
        print('Set value of lower priority')
        item.get_mode('automatic').set_value(55)
        print('Disable higher priority')
        item.get_mode('manual').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
```

10.3.2 Advanced Example

```
import logging
import HABApp
from HABApp.core.events import ValueUpdateEvent
from HABApp.util import MultiModeItem

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

(continues on next page)

(continued from previous page)

```
        # create a new MultiModeItem and assign logger
        log = logging.getLogger('AdvancedMultiMode')
        item = MultiModeItem.get_create_item('MultiModeTestItem', log)
        self.listen_event(item, self.item_update, ValueUpdateEvent)

        # create two different modes which we will use
        item.create_mode('Automatic', 2, initial_value=5)
        item.create_mode('Manual', 10).set_value(10)
        print(f'{repr(item.get_mode("Automatic"))}')
        print(f'{repr(item.get_mode("Manual"))}')


        # 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('')
        print('-' * 80)
        print('Automatically disable mode')
        print('-' * 80)
        item.get_mode('manual').auto_disable_on = '>='  # disable when low priority␣
↪value >= mode value

        item.get_mode('Automatic').set_value(11)    # <-- manual now gets disabled␣
↪because
        item.get_mode('Automatic').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␣
↪disable the
        # manual mode or if the state should be remembered then the max function␣
↪should be used
        print('')
        print('-' * 80)
        print('Use of functions')
        print('-' * 80)
        item.create_mode('Manual', 10, initial_value=5, calc_value_func=max)    #␣
↪overwrite the earlier declaration
        item.get_mode('Automatic').set_value(7)
        item.get_mode('Automatic').set_value(3)

    def item_update(self, event):
        print(f'State: {event.value}')

MyMultiModeItemTestRule()
```

```
[AdvancedMultiMode]     INFO | MultiModeTestItem: Manual set value to 10
State: 10
<MultiModeValue Automatic enabled: True, value: 5>
<MultiModeValue Manual enabled: True, value: 10>

--------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
Automatically disable mode
-------------------------------------------------------------------------------
[AdvancedMultiMode]     INFO | MultiModeTestItem: Automatic set value to 11
[AdvancedMultiMode]     INFO | MultiModeTestItem: Manual disabled (11>=10)!
State: 11
[AdvancedMultiMode]     INFO | MultiModeTestItem: Automatic set value to 4
State: 4

-------------------------------------------------------------------------------
Use of functions
-------------------------------------------------------------------------------
[AdvancedMultiMode]     INFO | MultiModeTestItem: Automatic set value to 7
State: 7
[AdvancedMultiMode]     INFO | MultiModeTestItem: Automatic set value to 3
State: 5
```

10.3.3 Documentation

class HABApp.util.**MultiModeItem**(*name*, *initial_value=None*)

Thread safe value prioritizer *Item*

Variables **logger** – Assign a logger to get log messages about the different modes

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

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

Parameters

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

Returns item

create_mode(*name*, *priority*, *initial_value=None*, *auto_disable_on=None*, *auto_disable_after=None*, *calc_value_func=None*)

Create a new mode with priority

Parameters

- **name** (str) – Name of the new mode
- **priority** (int) – Priority of the mode
- **initial_value** (Optional[Any]) – Initial value, will also enable the mode
- **auto_disable_on** (Optional[str]) – Automatically disable the mode if the lower priority state is > or < the value. See *MultiModeValue*
- **auto_disable_after** (Optional[timedelta]) – Automatically disable the mode after a timedelta if a recalculate is done See *MultiModeValue*
- **calc_value_func** (Optional[Callable[[Any, Any], Any]]) – See *MultiModeValue*

Return type *MultiModeValue*

Returns The newly created MultiModeValue

get_mode(*name*)

Returns a created mode

Parameters **`name`** (`str`) – name of the mode (case insensitive)

Return type *`MultiModeValue`*

Returns The requested MultiModeValue

`calculate_value``()`
Recalculate the output value and post the state to the event bus (if it is not None)

Return type `Any`

Returns new value

class `HABApp.util.multimode_item.`**`MultiModeValue`**(*parent*, *name*, *initial_value=None*, *auto_disable_on=None*, *auto_disable_after=None*, *calc_value_func=None*)

Variables

- **`last_update`** (*`datetime.datetime`*) – Timestamp of the last update/enable of this value
- **`auto_disable_after`** (*`typing.Optional[datetime.timedelta]`*) – Automatically disable this mode after a given timedelta on the next recalculation
- **`auto_disable_on`** (*`typing.Optional[str]`*) – Automatically disable this mode if the state with lower priority is >, >=, <, <=, == or != than the own value
- **`calc_value_func`** (*`typing.Optional[typing.Callable[[typing.Any, typing.Any], typing.Any]]`*) – Function to calculate the new value (e.g. `min` or `max`). Any function that accepts two Arguments can be used. First arg is value with lower priority, second argument is own value.

`property value`
Returns the current value

`property enabled`
Returns if the value is enabled

Return type `bool`

`set_value`(*value*)
Set new value and recalculate overall value

Parameters **`value`** – new value

`set_enabled`(*value*)
Enable or disable this value and recalculate overall value

Parameters **`value`** (`bool`) – True/False

CHAPTER

ELEVEN

ADDITIONAL RULE EXAMPLES

11.1 Using the scheduler

```python
import datetime
import time

import HABApp


class MyRule(HABApp.Rule):

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

        self.run_on_day_of_week(
            datetime.time(14, 34, 20),
            weekdays=['Mo'],
            callback=self.run_mondays
        )

        self.run_every(datetime.timedelta(seconds=5), 3, self.run_every_3s, 'arg 1',␣
↪asdf='kwarg 1')

        self.run_on_workdays(datetime.time( 15, 00), self.run_workdays)
        self.run_on_weekends(datetime.time( 15, 00), self.run_weekends)

    def run_every_3s(self, arg, asdf = None):
        print( f'run_ever_3s: {time.time():.3f} : {arg}, {asdf}')

    def run_mondays(self):
        print('Today is monday!')

    def run_workdays(self):
        print('Today is a workday!')

    def run_weekends(self):
        print('Today is weekend!')


MyRule()
```

11.2 Mirror openHAB events to a MQTT Broker

```
import HABApp
from HABApp.openhab.events import ItemStateEvent


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

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

        self.listen_event(None, self.process_any_update, ItemStateEvent)

    def process_any_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()
```

11.3 Trigger an event when an item is constant

```
import HABApp
from HABApp.core.events import ValueNoChangeEvent

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

        self.item_watch('test_watch', 5)
        self.item_watch('test_watch', 10)
        self.listen_event('test_watch', self.item_constant, ValueNoChangeEvent)

        self.set_item_state('test_watch', 'my_value')

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

MyRule()
```

```
<ValueNoChangeEvent name: test_watch, value: my_value, seconds: 5>
<ValueNoChangeEvent name: test_watch, value: my_value, seconds: 10>
-c:18: DeprecationWarning:'set_item_state' is deprecated, use 'post_value' or 'set_
↪value' method of the item instance instead
```

11.4 Process Errors in Rules

This example shows how to create a rule with a function which will be called when **any** rule throws an error. The rule function then can push the error message to an openhab item or e.g. use Pushover to send the error message to the mobile device (see *Avanced Usage* for more information).

```python
import HABApp
from HABApp.core.events.habapp_events import HABAppError

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

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

    def on_error(self, error_event: HABAppError):
        print(error_event.to_str())

NotifyOnError()


# this is a faulty example. Do not create this part!
class FaultyRule(HABApp.Rule):
    def __init__(self):
        super().__init__()
        self.run_soon(self.faulty_function)

    def faulty_function(self):
        1 / 0
FaultyRule()
```

```
Error in FaultyRule.faulty_function: division by zero
Traceback (most recent call last):
  File "<string>", line 29, in faulty_function
ZeroDivisionError: division by zero
```

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

INDEX