

Airgonimic\_Backend

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Namespace Documentation</b>	<b>5</b>
3.1	backend.resources.configuration Namespace Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	backend.resources.information Namespace Reference . . . . .	5
3.2.1	Detailed Description . . . . .	5
3.3	backend.resources.login Namespace Reference . . . . .	5
3.3.1	Detailed Description . . . . .	5
3.4	backend.resources.register Namespace Reference . . . . .	5
3.4.1	Detailed Description . . . . .	5
3.5	backend.resources.room Namespace Reference . . . . .	6
3.5.1	Detailed Description . . . . .	6
3.6	backend.resources.user Namespace Reference . . . . .	6
3.6.1	Detailed Description . . . . .	6
3.7	backend.resources.window Namespace Reference . . . . .	6
3.7.1	Detailed Description . . . . .	6
3.8	backend.util.__init__ Namespace Reference . . . . .	6
3.8.1	Detailed Description . . . . .	6
3.9	backend.util.arg_parser Namespace Reference . . . . .	6

3.9.1 Detailed Description . . . . .	6
3.10 backend.util.db Namespace Reference . . . . .	6
3.10.1 Detailed Description . . . . .	6
3.11 backend.util.response_code Namespace Reference . . . . .	7
3.11.1 Detailed Description . . . . .	7
3.12 backend.util.security Namespace Reference . . . . .	7
3.12.1 Detailed Description . . . . .	7
3.13 control.__init__ Namespace Reference . . . . .	7
3.13.1 Detailed Description . . . . .	7
3.14 control.configuration_handler Namespace Reference . . . . .	7
3.14.1 Detailed Description . . . . .	7
3.14.2 Function Documentation . . . . .	8
3.14.2.1 load_configuration() . . . . .	8
3.14.2.2 parse_file() . . . . .	8
3.14.2.3 save_configuration() . . . . .	8
3.14.3 Variable Documentation . . . . .	9
3.14.3.1 filename . . . . .	9
3.15 control.led_handler Namespace Reference . . . . .	9
3.15.1 Detailed Description . . . . .	9
3.15.2 Function Documentation . . . . .	9
3.15.2.1 close_window() . . . . .	9
3.15.2.2 open_window() . . . . .	10
3.16 control.room Namespace Reference . . . . .	10
3.16.1 Detailed Description . . . . .	10
3.16.2 Function Documentation . . . . .	10
3.16.2.1 register_new_room() . . . . .	10
3.17 control.sensors.read_bme280_hum Namespace Reference . . . . .	11
3.17.1 Detailed Description . . . . .	11
3.18 control.sensors.read_t6613_co2 Namespace Reference . . . . .	11
3.18.1 Detailed Description . . . . .	11
3.19 control.window Namespace Reference . . . . .	11
3.19.1 Detailed Description . . . . .	11
3.19.2 Function Documentation . . . . .	11
3.19.2.1 register_new_window() . . . . .	11

<b>4</b>	<b>Class Documentation</b>	<b>13</b>
4.1	control.room.Room Class Reference	13
4.1.1	Detailed Description	14
4.1.2	Constructor & Destructor Documentation	14
4.1.2.1	__init__()	14
4.1.3	Member Function Documentation	14
4.1.3.1	__str__()	14
4.1.3.2	check_status()	15
4.1.3.3	close_all()	15
4.1.3.4	get_configuration()	15
4.1.3.5	get_id()	16
4.1.3.6	get_threshold()	16
4.1.3.7	get_update()	16
4.1.3.8	get_window_count()	18
4.1.3.9	set_update()	18
4.2	control.window.Window Class Reference	18
4.2.1	Detailed Description	19
4.2.2	Constructor & Destructor Documentation	19
4.2.2.1	__init__()	19
4.2.3	Member Function Documentation	20
4.2.3.1	__str__()	20
4.2.3.2	change_state()	20
4.2.3.3	get_configuration()	21
4.2.3.4	get_update()	21
4.2.3.5	set_update()	21
	<b>Index</b>	<b>23</b>



# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">backend.resources.configuration</a>	5
<a href="#">backend.resources.information</a>	5
<a href="#">backend.resources.login</a>	5
<a href="#">backend.resources.register</a>	5
<a href="#">backend.resources.room</a>	6
<a href="#">backend.resources.user</a>	6
<a href="#">backend.resources.window</a>	6
<a href="#">backend.util.__init__</a>	6
<a href="#">backend.util.arg_parser</a>	6
<a href="#">backend.util.db</a>	6
<a href="#">backend.util.response_code</a>	7
<a href="#">backend.util.security</a>	7
<a href="#">control.__init__</a>	7
<a href="#">control.configuration_handler</a>	7
<a href="#">control.led_handler</a>	9
<a href="#">control.room</a>	10
<a href="#">control.sensors.read_bme280_hum</a>	11
<a href="#">control.sensors.read_t6613_co2</a>	11
<a href="#">control.window</a>	11





## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">control.room.Room</a>	
Handles the functionality of a room . . . . .	13
<a href="#">control.window.Window</a>	
Handles the functionality of a window . . . . .	18



## Chapter 3

# Namespace Documentation

### 3.1 `backend.resources.configuration` Namespace Reference

#### 3.1.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the configuration resources. See rest api documentation for further information.

### 3.2 `backend.resources.information` Namespace Reference

#### 3.2.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the information resources. See rest api documentation for further information.

### 3.3 `backend.resources.login` Namespace Reference

#### 3.3.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the login resources. See rest api documentation for further information.

### 3.4 `backend.resources.register` Namespace Reference

#### 3.4.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the register resources. See rest api documentation for further information.

### 3.5 `backend.resources.room` Namespace Reference

#### 3.5.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the room ressources. See rest api documentation for further information.

### 3.6 `backend.resources.user` Namespace Reference

#### 3.6.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the user ressources. See rest api documentation for further information.

### 3.7 `backend.resources.window` Namespace Reference

#### 3.7.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the window ressources. See rest api documentation for further information.

### 3.8 `backend.util.__init__` Namespace Reference

#### 3.8.1 Detailed Description

##### Author

Sebastian Steinmeyer The application factory of the backend

### 3.9 `backend.util.arg_parser` Namespace Reference

#### 3.9.1 Detailed Description

##### Author

Sebastian Steinmeyer Offers the functinallity to simple parse integer and float within defined range.

### 3.10 `backend.util.db` Namespace Reference

#### 3.10.1 Detailed Description

##### Author

Sebastian Steinmeyer Handles the functionality for the database access

## 3.11 backend.util.response\_code Namespace Reference

### 3.11.1 Detailed Description

#### Author

Sebastian Steinmeyer Offers constants for the http status codes used in the backend.

## 3.12 backend.util.security Namespace Reference

### 3.12.1 Detailed Description

#### Author

Sebastian Steinmeyer Handles the checks for user rights and getting room/user by security token. At the moment the token is simply the id of the user/room.

## 3.13 control.\_\_init\_\_ Namespace Reference

### 3.13.1 Detailed Description

#### Author

Sebastian Steinmeyer Handles the start, close and runtime of the window control.

## 3.14 control.configuration\_handler Namespace Reference

### Functions

- def `parse_file` (columns)  
*Read the configuration as a map with lists from file.*
- def `load_configuration` ()  
*Parse the configuration and create rooms and windows based on it.*
- def `save_configuration` (sleep\_duration, room)  
*Saves the configuration into configuration file.*

### Variables

- string `filename` = 'config.txt'  
*The filename of the configuration file.*

### 3.14.1 Detailed Description

#### Author

Sebastian Steinmeyer Handles the load and save of configurations from file.

### 3.14.2 Function Documentation

#### 3.14.2.1 load\_configuration()

```
def control.configuration_handler.load_configuration ( )
```

Parse the configuration and create rooms and windows based on it.

The values backend, window\_count, sleep\_duration and the window gpios in form of window\_i,[gpio] need to be defined in the configuration file. In case room is not given, a new room with given windows will be created. Sample structure of the configuration file: backend,<ip:port>:int,None sleep\_duration,<duration>:int,None window\_count,<count>:int,None room,<id in="" the="" backend>="">:int,None window\_i,<is in="" the="" backend>="">:int,<gpio>:int

##### Returns

the parsed Room with the parsed Windows

#### 3.14.2.2 parse\_file()

```
def control.configuration_handler.parse_file (
    columns )
```

Read the configuration as a map with lists from file.

The separator is defined to ",".

##### Parameters

<i>columns</i>	the number of columns to read in each row.
----------------	--

##### Returns

a dictionary with a list of parsed values

#### 3.14.2.3 save\_configuration()

```
def control.configuration_handler.save_configuration (
    sleep_duration,
    room )
```

Saves the configuration into configuration file.

## Parameters

<i>sleep_duration</i>	the defined duration
<i>room</i>	the room of type Room to get the configuration from

### 3.14.3 Variable Documentation

#### 3.14.3.1 filename

```
string control.configuration_handler.filename = 'config.txt'
```

The filename of the configuration file.

## 3.15 control.led\_handler Namespace Reference

## Functions

- def [open\\_window](#) (gpio)  
*Turn on a led.*
- def [close\\_window](#) (gpio)  
*Turn off a led.*

### 3.15.1 Detailed Description

## Author

Sebastian Steinmeyer Handles the functionality to turn a led on and off.

### 3.15.2 Function Documentation

#### 3.15.2.1 close\_window()

```
def control.led_handler.close_window (  
    gpio )
```

Turn off a led.

## Parameters

<i>gpio</i>	the gpoi pin which should turn off.
-------------	-------------------------------------

### 3.15.2.2 open\_window()

```
def control.led_handler.open_window (  
    gpio )
```

Turn on a led.

#### Parameters

<i>gpio</i>	the gpoi pin which should turn on.
-------------	------------------------------------

## 3.16 control.room Namespace Reference

### Classes

- class [Room](#)  
*Handles the functionality of a room.*

### Functions

- def [register\\_new\\_room](#) (backend)  
*Registers a new room at the given backend.*

### 3.16.1 Detailed Description

#### Author

Sebastian Steinmeyer Handles the class [Room](#) and a funtion to register new rooms.

### 3.16.2 Function Documentation

#### 3.16.2.1 register\_new\_room()

```
def control.room.register_new_room (  
    backend )
```

Registers a new room at the given backend.

#### Parameters

<i>backend</i>	the ip and port of the target backend as string
----------------	---



**Returns**

the created [Room](#) object or None in case of error

## 3.17 control.sensors.read\_bme280\_hum Namespace Reference

### 3.17.1 Detailed Description

**Author**

Sebastian Steinmeyer Handles the functionality to read the bme280 sensor

## 3.18 control.sensors.read\_t6613\_co2 Namespace Reference

### 3.18.1 Detailed Description

**Author**

Sebastian Steinmeyer Handles the functionality to read the t6613 sensor

## 3.19 control.window Namespace Reference

**Classes**

- class [Window](#)  
*Handles the functionality of a window.*

**Functions**

- def [register\\_new\\_window](#) (backend, room\_id, gpio)  
*Registers a new window at the given backend.*

### 3.19.1 Detailed Description

**Author**

Sebastian Steinmeyer Handles the class [Window](#) and a function to register new windows.

### 3.19.2 Function Documentation

#### 3.19.2.1 register\_new\_window()

```
def control.window.register_new_window (
    backend,
    room_id,
    gpio )
```

Registers a new window at the given backend.

**Parameters**

<i>backend</i>	the ip and port of the target backend as string
<i>room↔ _id</i>	the id of the window containing room
<i>gpio</i>	the gpio pin of the used led

**Returns**

the created [Window](#) object or None in case of error

## Chapter 4

# Class Documentation

### 4.1 control.room.Room Class Reference

Handles the functionality of a room.

#### Public Member Functions

- def `__init__` (self, id, backend)  
*Standart constructor to initiate a room with default values.*
- def `__str__` (self)  
*Get the room as string representation.*
- def `get_id` (self)  
*Get the room id.*
- def `get_update` (self)  
*Updates the room with the data from the given backend.*
- def `set_update` (self)  
*Pushes the room values to the given backend.*
- def `add_window` (self, window)
- def `check_status` (self)  
*Checks the air values and decide to open the windows or not.*
- def `get_threshold` (self)  
*Get the threshold from the given backend.*
- def `get_configuration` (self)  
*Get the configuration of the room and its windows.*
- def `get_window_count` (self)  
*Get the count of containing windows.*
- def `close_all` (self)  
*Closes all containing windows.*

## Public Attributes

- **id**
- **backend\_raw**
- **backend**
- **co2**
- **humidity**
- **automatic**
- **open**
- **force**
- **windows**
- **override**

### 4.1.1 Detailed Description

Handles the functionality of a room.

This include getting and setting updates and check the air values.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 `__init__()`

```
def control.room.Room.__init__ (
    self,
    id,
    backend )
```

Standart constructor to initiate a room with default values.

Please call `get_update` after creating a room.

#### Parameters

<i>self</i>	the object pointer
<i>id</i>	the room id in the backend
<i>backend</i>	the ip and port of the used backend

### 4.1.3 Member Function Documentation

#### 4.1.3.1 `__str__()`

```
def control.room.Room.__str__ (
    self )
```

Get the room as string representation.

#### Parameters

<i>self</i>	the object pointer
-------------	--------------------

#### Returns

the room as string

#### 4.1.3.2 check\_status()

```
def control.room.Room.check_status (
    self )
```

Checks the air values and decide to open the windows or not.

In case one value is over threshold it calls open for all windows. It contains an override, which lasts till the user change it.

#### Parameters

<i>self</i>	the object pointer
-------------	--------------------

#### 4.1.3.3 close\_all()

```
def control.room.Room.close_all (
    self )
```

Closes all containing windows.

#### Parameters

<i>self</i>	the object pointer
-------------	--------------------

#### 4.1.3.4 get\_configuration()

```
def control.room.Room.get_configuration (
    self )
```

Get the configuration of the room and its windows.

**Parameters**

<i>self</i>	the object pointer
-------------	--------------------

**Returns**

the configuration as string

**4.1.3.5 get\_id()**

```
def control.room.Room.get_id (
    self )
```

Get the room id.

**Parameters**

<i>self</i>	the object pointer
-------------	--------------------

**Returns**

the id of the room

**4.1.3.6 get\_threshold()**

```
def control.room.Room.get_threshold (
    self )
```

Get the threshold from the given backend.

**Parameters**

<i>self</i>	the object pointer.
-------------	---------------------

**Returns**

a dictionary with the received values.

**4.1.3.7 get\_update()**

```
def control.room.Room.get_update (
    self )
```

Updates the room with the data from the given backend.

**Parameters**

<i>self</i>	the object pointer.
-------------	---------------------

**4.1.3.8 get\_window\_count()**

```
def control.room.Room.get_window_count (
    self )
```

Get the count of containing windows.

**Parameters**

<i>self</i>	the object pointer
-------------	--------------------

**Returns**

the window count as integer

**4.1.3.9 set\_update()**

```
def control.room.Room.set_update (
    self )
```

Pushes the room values to the given backend.

**Parameters**

<i>self</i>	the object pointer
-------------	--------------------

The documentation for this class was generated from the following file:

- control/room.py

## 4.2 control.window.Window Class Reference

Handles the functionality of a window.



## Public Member Functions

- `def __init__ (self, id, room_id, backend, gpio)`  
*Standart constructor to initiate a window with default values.*
- `def __str__ (self)`  
*Get the window as string representation.*
- `def get_update (self)`  
*Updates the window with the data from the given backend.*
- `def set_update (self)`  
*Pushes the window values to the given backend.*
- `def change_state (self, to_state)`  
*Changes the window into the given state.*
- `def get_configuration (self, index)`  
*Get the configuration of the window.*

## Public Attributes

- `id`
- `room_id`
- `backend`
- `gpio`
- `automatic`
- `open`

### 4.2.1 Detailed Description

Handles the functionality of a window.

This include getting and setting updates and open and close it.

### 4.2.2 Constructor & Destructor Documentation

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

```
def control.window.Window.__init__ (
    self,
    id,
    room_id,
    backend,
    gpio )
```

Standart constructor to initiate a window with default values.

Please call `get_update` after creating a window.

**Parameters**

<i>self</i>	the object pointer
<i>id</i>	the window id in the backend
<i>room↔ _id</i>	the id of the room which contains the window
<i>backend</i>	the ip and port of the used backend
<i>gpio</i>	the gpio pin for the led which simbolize the window

**4.2.3 Member Function Documentation****4.2.3.1 \_\_str\_\_()**

```
def control.window.Window.__str__ (
    self )
```

Get the window as string representation.

**Parameters**

<i>self</i>	the object pointer
-------------	--------------------

**Returns**

the window as string

**4.2.3.2 change\_state()**

```
def control.window.Window.change_state (
    self,
    to_state )
```

Changes the window into the given state.

If the state is already set or automatic is disabled, nothing will happen.

**Parameters**

<i>self</i>	the object pointer
<i>to_state</i>	the targeted state

#### 4.2.3.3 get\_configuration()

```
def control.window.Window.get_configuration (
    self,
    index )
```

Get the configuration of the window.

##### Parameters

<i>self</i>	the object pointer
<i>index</i>	the window index related to the room

##### Returns

the configuration as string

#### 4.2.3.4 get\_update()

```
def control.window.Window.get_update (
    self )
```

Updates the window with the data from the given backend.

##### Parameters

<i>self</i>	the object pointer.
-------------	---------------------

#### 4.2.3.5 set\_update()

```
def control.window.Window.set_update (
    self )
```

Pushes the window values to the given backend.

##### Parameters

<i>self</i>	the object pointer
-------------	--------------------

The documentation for this class was generated from the following file:

- control/window.py



# Index

- `__init__`
    - `control::room::Room`, 14
    - `control::window::Window`, 19
  - `__str__`
    - `control::room::Room`, 14
    - `control::window::Window`, 20
- `backend.resources.configuration`, 5
- `backend.resources.information`, 5
- `backend.resources.login`, 5
- `backend.resources.register`, 5
- `backend.resources.room`, 6
- `backend.resources.user`, 6
- `backend.resources.window`, 6
- `backend.util.__init__`, 6
- `backend.util.arg_parser`, 6
- `backend.util.db`, 6
- `backend.util.response_code`, 7
- `backend.util.security`, 7
- `change_state`
  - `control::window::Window`, 20
- `check_status`
  - `control::room::Room`, 15
- `close_all`
  - `control::room::Room`, 15
- `close_window`
  - `control::led_handler`, 9
- `control.__init__`, 7
- `control.configuration_handler`, 7
- `control.led_handler`, 9
- `control.room`, 10
- `control.room.Room`, 13
- `control.sensors.read_bme280_hum`, 11
- `control.sensors.read_t6613_co2`, 11
- `control.window`, 11
- `control.window.Window`, 18
- `control::configuration_handler`
  - `filename`, 9
  - `load_configuration`, 8
  - `parse_file`, 8
  - `save_configuration`, 8
- `control::led_handler`
  - `close_window`, 9
  - `open_window`, 10
- `control::room`
  - `register_new_room`, 10
- `control::room::Room`
  - `__init__`, 14
  - `__str__`, 14
  - `check_status`, 15
  - `close_all`, 15
  - `get_configuration`, 15
  - `get_id`, 16
  - `get_threshold`, 16
  - `get_update`, 16
  - `get_window_count`, 18
  - `set_update`, 18
- `control::window`
  - `register_new_window`, 11
- `control::window::Window`
  - `__init__`, 19
  - `__str__`, 20
  - `change_state`, 20
  - `get_configuration`, 20
  - `get_update`, 21
  - `set_update`, 21
- `filename`
  - `control::configuration_handler`, 9
- `get_configuration`
  - `control::room::Room`, 15
  - `control::window::Window`, 20
- `get_id`
  - `control::room::Room`, 16
- `get_threshold`
  - `control::room::Room`, 16
- `get_update`
  - `control::room::Room`, 16
  - `control::window::Window`, 21
- `get_window_count`
  - `control::room::Room`, 18
- `load_configuration`
  - `control::configuration_handler`, 8
- `open_window`
  - `control::led_handler`, 10
- `parse_file`
  - `control::configuration_handler`, 8
- `register_new_room`
  - `control::room`, 10
- `register_new_window`
  - `control::window`, 11
- `save_configuration`
  - `control::configuration_handler`, 8
- `set_update`

control::room::Room, [18](#)  
control::window::Window, [21](#)