bimmer_connected Documentation

m1n3rva

README

1	Installation	3				
2	Usage 2.1 Example in an asyncio event loop	5 5				
3	Compatibility	7				
4	Data Contributions					
5	5 Code Contributions					
6	Thank you					
7	License	15				
8	Disclaimer 8.1 bimmer_connected 8.2 Installation 8.3 Usage 8.4 Compatibility 8.5 Data Contributions 8.6 Code Contributions 8.7 Thank you 8.8 License 8.9 Disclaimer 8.10 Using fingerprints in Home Assistant 8.11 Reverse engineering the MyBMW API 8.12 bimmer_connected 8.13 bimmer_connected. 8.14 bimmer_connected.account 8.14 bimmer_connected.api 8.15 bimmer_connected.models 8.16 bimmer_connected.utils 8.17 bimmer_connected.vehicle	177 177 188 199 200 200 200 211 244 300 322 344 355 388				
9	Indices and tables					
Py	Python Module Index					
Inc	ndex 5					



This is a simple library to query and control the status of your BMW or Mini vehicle from the MyBMW portal.

README 1

2 README

ONE

INSTALLATION

bimmer_connected is tested against **Python 3.8 or above**. Just install the latest release from PyPI using pip3 install --upgrade bimmer_connected.

Alternatively, clone the project and execute pip install -e . to install the current master branch.

Note: If you want to connect to a **chinese** server, you need to install the [china] extra, e.g. pip3 install --upgrade bimmer_connected[china].

TWO

USAGE

While this library is mainly written to be included in Home Assistant, it can be use on its own.

After installation, execute bimmerconnected from command line for usage instruction or see the full CLI documentation.

Please be aware that bimmer_connected is an async library when using it in Python code. The description of the modules can be found in the module documentation.

2.1 Example in an asyncio event loop

```
import asyncio
from bimmer_connected.account import MyBMWAccount
from bimmer_connected.api.regions import Regions

async def main():
    account = MyBMWAccount(USERNAME, PASSWORD, Regions.REST_OF_WORLD)
    await account.get_vehicles()
    vehicle = account.get_vehicle(VIN)
    print(vehicle.brand, vehicle.name, vehicle.vin)

    result = await vehicle.remote_services.trigger_remote_light_flash()
    print(result.state)

asyncio.run(main())
```

2.2 Example in non-async code

```
import asyncio
from bimmer_connected.account import MyBMWAccount
from bimmer_connected.api.regions import Regions

account = MyBMWAccount(USERNAME, PASSWORD, Regions.REST_OF_WORLD)
asyncio.run(account.get_vehicles())
vehicle = account.get_vehicle(VIN)
print(vehicle.brand, vehicle.name, vehicle.vin)
```

(continues on next page)

(continued from previous page)

result = asyncio.run(vehicle.remote_services.trigger_remote_light_flash())
print(result.state)

6 Chapter 2. Usage

THREE

COMPATIBILITY

This works with BMW (and Mini) vehicles with a MyBMW account. So far it is tested on vehicles with a 'MGU', 'NBTEvo', 'EntryEvo', 'NBT', or 'EntryNav' navigation system. If you have any trouble with other navigation systems, please create an issue with your server responses (see next section).

To use this library, your BMW (or Mini) must have the remote services enabled for your vehicle. You might need to book this in the MyBMW/Mini Connected portal and this might cost some money. In addition to that you need to enable the Remote Services in your infotainment system in the vehicle.

Different models of vehicles and infotainment systems result in different types of attributes provided by the server. So the experience with the library will certainly vary across the different vehicle models.

FOUR

DATA CONTRIBUTIONS

If some features do not work for your vehicle, we would need the data returned form the server to analyse this and potentially extend the code. Different models and head unit generations lead to different responses from the server.

If you want to contribute your data, perform the following steps:

```
# get the latest version of the library
pip3 install --upgrade bimmer_connected

# run the fingerprint function
bimmerconnected fingerprint <username> <password> <region>
```

This will create a set of log files in the "vehicle_fingerprint" folder. Before sending the data to anyone please **check for any personal data** such as **dealer name** or **country**.

The following attributes are by default replaced with anonymized values:

- vin (Vehicle Identification Number)
- lat and lon (GPS position)
- licensePlate
- information of dealer

Create a new fingerprint data contribution and add the files as attachment to the discussion.

Please add your model and year to the title of the issue, to make it easier to organize. If you know the "chassis code" of your car, you can include that too. (For example, googling "2017 BMW X5" will show a Wikipedia article entitled "BMW X5 (F15)". F15 is therefore the chassis code of the car.)

Note: We will then use this data as additional test cases. So we will publish (parts of) it (after checking for personal information again) and use this as test cases for our library. If you do not want this, please let us know in advance.

FIVE

CODE CONTRIBUTIONS

Contributions are welcome! Please make sure that your code passes the checks in .github/workflows/test.yml. We currently test against flake8, pylint and our own pytest suite. And please add tests where it makes sense. The more the better.

See the contributing guidelines for more details.

SIX

THANK YOU

Thank you to all contributors for your research and contributions! And thanks to everyone who shares the fingerprint data of their vehicles which we use to test the code. A special thanks to @HuChundong, @muxiachuixue, @vividmuse for figuring out how to solve login issues!

This library is basically a best-of of other similar solutions, yet none of them provided a ready to use library with a matching interface to be used in Home Assistant and is available on pypi.

/

- https://github.com/edent/BMW-i-Remote
- https://github.com/jupe76/bmwcdapi
- https://github.com/frankjoke/iobroker.bmw
- https://github.com/TA2k/ioBroker.bmw
- https://gitee.com/ichuixue/bmw_shortcuts eb064e89e6b647d2828a404227b91c4a

https://www.icloud.com/shortcuts/

Thank you for your great software!

CHAPTER SEVEN

LICENSE

The bimmer_connected library is licensed under the Apache License 2.0.

16 Chapter 7. License

EIGHT

DISCLAIMER

This library is not affiliated with or endorsed by BMW Group.

8.1 bimmer_connected

This is a simple library to query and control the status of your BMW or Mini vehicle from the MyBMW portal.

8.2 Installation

bimmer_connected is tested against **Python 3.8 or above**. Just install the latest release from PyPI using pip3 install --upgrade bimmer_connected.

Alternatively, clone the project and execute pip install -e . to install the current master branch.

Note: If you want to connect to a **chinese** server, you need to install the [china] extra, e.g. pip3 install --upgrade bimmer_connected[china].

8.3 Usage

While this library is mainly written to be included in Home Assistant, it can be use on its own.

After installation, execute bimmerconnected from command line for usage instruction or see the full CLI documentation.

Please be aware that bimmer_connected is an async library when using it in Python code. The description of the modules can be found in the module documentation.

8.3.1 Example in an asyncio event loop

```
import asyncio
from bimmer_connected.account import MyBMWAccount
from bimmer_connected.api.regions import Regions

async def main():
    account = MyBMWAccount(USERNAME, PASSWORD, Regions.REST_OF_WORLD)
    await account.get_vehicles()
    vehicle = account.get_vehicle(VIN)
    print(vehicle.brand, vehicle.name, vehicle.vin)

    result = await vehicle.remote_services.trigger_remote_light_flash()
    print(result.state)

asyncio.run(main())
```

8.3.2 Example in non-async code

```
import asyncio
from bimmer_connected.account import MyBMWAccount
from bimmer_connected.api.regions import Regions

account = MyBMWAccount(USERNAME, PASSWORD, Regions.REST_OF_WORLD)
asyncio.run(account.get_vehicles())
vehicle = account.get_vehicle(VIN)
print(vehicle.brand, vehicle.name, vehicle.vin)

result = asyncio.run(vehicle.remote_services.trigger_remote_light_flash())
print(result.state)
```

8.4 Compatibility

This works with BMW (and Mini) vehicles with a MyBMW account. So far it is tested on vehicles with a 'MGU', 'NBTEvo', 'EntryEvo', 'NBT', or 'EntryNav' navigation system. If you have any trouble with other navigation systems, please create an issue with your server responses (see next section).

To use this library, your BMW (or Mini) must have the remote services enabled for your vehicle. You might need to book this in the MyBMW/Mini Connected portal and this might cost some money. In addition to that you need to enable the Remote Services in your infotainment system in the vehicle.

Different models of vehicles and infotainment systems result in different types of attributes provided by the server. So the experience with the library will certainly vary across the different vehicle models.

8.5 Data Contributions

If some features do not work for your vehicle, we would need the data returned form the server to analyse this and potentially extend the code. Different models and head unit generations lead to different responses from the server.

If you want to contribute your data, perform the following steps:

```
# get the latest version of the library
pip3 install --upgrade bimmer_connected

# run the fingerprint function
bimmerconnected fingerprint <username> <password> <region>
```

This will create a set of log files in the "vehicle_fingerprint" folder. Before sending the data to anyone please **check for any personal data** such as **dealer name** or **country**.

The following attributes are by default replaced with anonymized values:

- vin (Vehicle Identification Number)
- lat and lon (GPS position)
- licensePlate
- information of dealer

Create a new fingerprint data contribution and add the files as attachment to the discussion.

Please add your model and year to the title of the issue, to make it easier to organize. If you know the "chassis code" of your car, you can include that too. (For example, googling "2017 BMW X5" will show a Wikipedia article entitled "BMW X5 (F15)". F15 is therefore the chassis code of the car.)

Note: We will then use this data as additional test cases. So we will publish (parts of) it (after checking for personal information again) and use this as test cases for our library. If you do not want this, please let us know in advance.

8.4. Compatibility 19

8.6 Code Contributions

Contributions are welcome! Please make sure that your code passes the checks in .github/workflows/test.yml. We currently test against flake8, pylint and our own pytest suite. And please add tests where it makes sense. The more the better.

See the contributing guidelines for more details.

8.7 Thank you

Thank you to all contributors for your research and contributions! And thanks to everyone who shares the fingerprint data of their vehicles which we use to test the code. A special thanks to @HuChundong, @muxiachuixue, @vividmuse for figuring out how to solve login issues!

This library is basically a best-of of other similar solutions, yet none of them provided a ready to use library with a matching interface to be used in Home Assistant and is available on pypi.

- https://github.com/edent/BMW-i-Remote
- https://github.com/jupe76/bmwcdapi
- · https://github.com/frankjoke/iobroker.bmw
- https://github.com/TA2k/ioBroker.bmw
- https://gitee.com/ichuixue/bmw_shortcuts eb064e89e6b647d2828a404227b91c4a

https://www.icloud.com/shortcuts/

Thank you for your great software!

8.8 License

The bimmer_connected library is licensed under the Apache License 2.0.

8.9 Disclaimer

This library is not affiliated with or endorsed by BMW Group.

8.10 Using fingerprints in Home Assistant

Sometimes it can be useful to load the **fingerprints** used for our **pytest suite** in the development of the Home Assistant component. This enables debugging of the UI in Home Assistant which is not possible from pytest alone.

Warning: This is for the Home Assistant development environment only! Do not do this on your live instance!

Setup and start Home Assistant in the development environment at least once and let all python packages install (hass -c ./config). If not already done, set up the **BMW Connected Drive Integration**. You need to login a MyBMW account at least once. Shut down Homeassistant afterwards.

Note: The MyBMW account does not need to contain vehicles, a demo account without attached vehicles is sufficient.

Now, we have to "hack" our mocked backend calls into Home Assistant.

Edit homeassistant/components/bmw_connected_drive/coordinator.py and locate the function def _async_update_data(). We now have to replace await self.account.get_vehicles(). The try .. except block should look like this:

As the first parameter, you can specify a list of VINs for debugging or leave it empty (None or []) to load all vehicles of our test suite.

8.11 Reverse engineering the MyBMW API

This document should be seen as a help in setting up a working environment to intercept traffic of the MyBMW app. Not every step will be described fully, this guide is rather a summary and list for further reading. It will most likely need adjustments to your specific setup.

The MyBMW app is built with the Flutter framework and needs some additional persuasion to reveal the traffic.

8.11.1 Disclaimer

Note that we are actively disabling important security measures such as SSL/TLS encryption to understand which commands and messages are shared between the MyBMW app and the MyBMW servers.

Also note that there could always be changes to the API or the app itself made by BMW to stop us from understanding what is going on.

8.11.2 Acknowledgement

Most of this document would not exist without the amazing work of Jeroen Becker:

- Intercepting traffic from Android Flutter applications (ARMv7)
- Intercepting Flutter traffic on Android (ARMv8)
- Intercepting Flutter traffic on iOS

8.11.3 Software & hardware requirements

Note: This document is based on the MyBMW **Android** app. It should work similarly using **iPhones**. If possible, please create a PR with more details.

You will need:

- · A proxy with MITM capabilities such as mitmproxy
- A **rooted** android phone with a version supported by MyBMW (currently Android 6.0 Marshmallow). It could also work using an Android emulator.
- Access to your phone using ADB (via USB)
- · ProxyDroid to forward all traffic to your proxy
- Ghidra to find the location to patch out SSL verification
- A python environment with frida
- frida-android-helper to help installing frida on your phone

8.11.4 Finding the location of SSL verification

The following steps are required if the location of the SSL verification function is not known. If it is, please continue with the *next section*. For more details, please refer to *Jeroen Becker's work*.

Get an APK/XAPK of the MyBMW app (from your phone or one of the many download sites). APK names include:

- de.bmw.connected.mobile20.cn(china)
- de.bmw.connected.mobile20.na (north america)
- de.bmw.connected.mobile20.row (rest of world)

Now extract config.arm64-v8a.apk or config.armeabi-v7a.apk from the APK package (depending of your phone's target architecture).

In Ghidra, load and analyze lib/ARCH/libflutter.so.

After analyze has finished, go to Search > For Scalar and search for value 390. Find mov r3, #0x186 and jump to it.

Double click on function name on right side to get the hex address and first bytes of the function

Example: 2d e9 f0 4f a3 b0 81 46 50 20 10 70

8.11.5 Preparations on phone

On your phone, add your custom CA certificates to the system store (instructions for emulator, but works on **rooted** devies in similar fashion). This is required as the login screen is using the default Android WebView component, which again behaves differently from Flutter (or rather, behaves like expected).

Add your local proxy server to your Android system using ProxyDroid.

8.11.6 Disabling SSL verification with frida

Install & upgrade frida-tools & frida-android-helper (see *requirements*). Make sure that both are on the latest version.

Create a frida hook named hook_flutter_disable_ssl.js with the following content. If needed, replace the search pattern and disable adding 0x01 on ARMv8.

```
function hook_ssl_verify_result(address)
  Interceptor.attach(address, {
   onEnter: function(args) {
     console.log("Disabling SSL validation")
   },
   onLeave: function(retval)
      console.log("Retval: " + retval)
     retval.replace(0x1);
   }
 });
function disablePinning()
var m = Process.findModuleByName("libflutter.so");
var pattern = "2d e9 f0 4f a3 b0 81 46 50 20 10 70" // MyBMW 1.5.1 to 1.7.0 (all regions)
var res = Memory.scan(m.base, m.size, pattern, {
  onMatch: function(address, size){
      console.log('[+] ssl_verify_result found at: ' + address.toString());
      // Add 0x01 because it's a THUMB function
      // Otherwise, we would get 'Error: unable to intercept function at 0x9906f8ac;
⇔please file a bug'
      // REQUIRED ON ARMv7 ONLY!!
     hook_ssl_verify_result(address.add(0x01));
   },
  onError: function(reason){
      console.log('[!] There was an error scanning memory');
   onComplete: function()
      console.log("All done")
   }
 });
setTimeout(disablePinning, 1000)
```

Connect to your phone via ADB with root permissions.

```
adb root && adb remount
```

Update & start frida server on the phone with frida-android-helper.

```
fah server update && fah server start
```

Start the MyBMW app from your computer via frida (adjust app identifier if needed).

```
frida -Uf de.bmw.connected.mobile20.row -l .\hook_flutter_disable_ssl.js --no-pause
```

Now you should be able to capture all traffic between your phone and the MyBMW API.

8.11.7 Using the information in bimmer_connected

If you learn anything by capturing the traffic, please create Issues/Feature Requests or Pull Requests to our repository. Information that should be included contains:

- The URL of the endpoint
- HTTP headers of your request (**DO NOT** include **Cookie** or **Authentication** headers)
- The request payload (if available)
- The request response (if available)

If the data contains personal information, please do not delete it but replace it with random data.

Warning: Double check if all information is **sanitized** and no personal information or authentication data is included.

8.12 bimmerconnected

A simple executable to use and test the library.

8.12.1 Positional Arguments

cmd Possible choices: status, fingerprint, lightflash, horn, vehiclefinder, chargingsettings, chargingprofile, charge, image, sendpoi_from_address

8.12.2 Named Arguments

--debug Print debug logs.

Default: False

8.12.3 Sub-commands

status

Get the current status of the vehicle.

bimmerconnected status [-h] [-j] [-v [VIN]]

username password {north_america,china,rest_of_world}

[lat] [lng]

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

lat (optional) Your current GPS latitude (as float)
lng (optional) Your current GPS longitude (as float)

Named Arguments

-j, --json Output as JSON only. Removes all other output.

Default: False

-v, --vin Output data for specified VIN only.

fingerprint

Save a vehicle fingerprint.

bimmerconnected fingerprint [-h]

username password

{north_america,china,rest_of_world} [lat] [lng]

8.12. bimmerconnected 25

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

lat (optional) Your current GPS latitude (as float)
lng (optional) Your current GPS longitude (as float)

lightflash

Flash the vehicle lights.

bimmerconnected lightflash [-h]

username password

{north_america,china,rest_of_world} vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

horn

Trigger the vehicle horn

bimmerconnected horn [-h]

username password {north_america,china,rest_of_world} vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

vehiclefinder

Update the vehicle GPS location.

bimmerconnected vehiclefinder [-h]

username password

{north_america,china,rest_of_world} vin [lat]

[lng]

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

lat (optional) Your current GPS latitude (as float)
lng (optional) Your current GPS longitude (as float)

chargingsettings

Set vehicle charging settings.

bimmerconnected chargingsettings [-h] [--target-soc [TARGET_SOC]]

[--ac-limit [AC_LIMIT]]

username password

{north_america,china,rest_of_world} vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

8.12. bimmerconnected 27

Named Arguments

--target-soc Desired charging target SoC

--ac-limit Maximum AC limit

chargingprofile

Set vehicle charging profile.

bimmerconnected chargingprofile [-h]

[--charging-mode [{IMMEDIATE_CHARGING,DELAYED_CHARGING}]]

[--precondition-climate [PRECONDITION_CLIMATE]]

username password

{north_america,china,rest_of_world} vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

Named Arguments

--charging-mode Possible choices: IMMEDIATE_CHARGING, DELAYED_CHARGING

Desired charging mode

--precondition-climate Precondition climate on charging windows

charge

Start/stop charging on enabled vehicles.

bimmerconnected charge [-h]

username password {north_america,china,rest_of_world}

vin {start,stop}

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number
action Possible choices: start, stop

image

Download a vehicle image.

bimmerconnected image [-h]

username password {north_america,china,rest_of_world}

vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

sendpoi

Send a point of interest to the vehicle.

bimmerconnected sendpoi [-h] [--name [NAME]] [--street [STREET]]

[--city [CITY]] [--postalcode [POSTALCODE]]

[--country [COUNTRY]]

username password {north_america,china,rest_of_world}

vin latitude longitude

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

latitude Latitude of the POI

8.12. bimmerconnected 29

longitude Longitude of the POI

Named Arguments

--name Name of the POI

Default: "Sent with by bimmer_connected"

--street (optional, display only) Street & House No. of the POI

--city (optional, display only) City of the POI

--postalcode (optional, display only) Postal code of the POI--country (optional, display only) Country of the POI

sendpoi from address

Send a point of interest parsed from a street address to the vehicle.

bimmerconnected sendpoi_from_address [-h] [-n [NAME]]

[-a ADDRESS [ADDRESS ...]]

username password

{north_america,china,rest_of_world} vin

Positional Arguments

usernameConnected Drive usernamepasswordConnected Drive password

region Possible choices: north_america, china, rest_of_world

Region of the Connected Drive account

vin Vehicle Identification Number

Named Arguments

-n, --name (optional, display only) Name of the POI
-a, --address (e.g. 'Street 17, city, zip, country')

8.13 bimmer_connected.account

Access to a MyBMW account and all vehicles therein.

class bimmer_connected.account.**MyBMWAccount**(username: str, password: dataclasses.InitVar[str], region:

Regions, config: MyBMWClientConfiguration = None, log_responses: dataclasses.InitVar[bool] = False, observer_position: dataclasses.InitVar[GPSPosition] =

None, use_metric_units:

dataclasses.InitVar[Optional[bool]] = None)

Create a new connection to the MyBMW web service.

```
async add_vehicle(vehicle\_base: dict, fetched\_at: datetime | None = None) <math>\rightarrow None
     Add or update a vehicle from the API responses.
config: MyBMWClientConfiguration = None
     Optional. If provided, username/password/region are ignored.
property gcid: str | None
     Returns the current GCID.
static get\_stored\_responses() \rightarrow List[AnonymizedResponse]
     Return responses stored if log_responses was set to True.
get\_vehicle(vin: str) \rightarrow MyBMWVehicle \mid None
     Get vehicle with given VIN.
     The search is NOT case sensitive. :param vin: VIN of the vehicle you want to get. :return: Returns None
     if no vehicle is found.
async get_vehicles(force\_init: bool = False) \rightarrow None
     Retrieve vehicle data from BMW servers.
log_responses: dataclasses.InitVar[bool] = False
     Optional. If set, all responses from the server will be logged to this directory.
observer_position: dataclasses.InitVar[GPSPosition] = None
     Optional. Required for getting a position on older cars.
password: dataclasses.InitVar[str]
     MyBMW password.
property refresh_token: str | None
     Returns the current refresh_token.
region: Regions
     Region of the account. See api. Regions.
set_observer_position(latitude: float, longitude: float) \rightarrow None
     Set the position of the observer for all vehicles.
set_refresh_token(refresh_token: str, gcid: str | None = None) \rightarrow None
     Overwrite the current value of the MyBMW refresh token and GCID (if available).
use_metric_units: dataclasses.InitVar[Optional[bool]] = None
     Deprecated. All returned values are metric units (km, 1).
username: str
     MyBMW user name (email) or 86-prefixed phone number (China only).
vehicles: List[MyBMWVehicle]
```

8.14 bimmer_connected.api

The bimmer_connected.api module contains helper functions to communicate with the BMW APIs.

8.14.1 bimmer_connected.api.authentication

Authentication management for BMW APIs.

class bimmer_connected.api.authentication.**MyBMWAuthentication**(username: str, password: str,

region: Regions, access_token: str | None = None, expires_at: datetime | None = None, refresh_token: str | None = None, gcid: str | None = None)

Authentication and Retry Handler for MyBMW API.

```
\textbf{async\_auth\_flow}(\textit{request: Request}) \rightarrow AsyncGenerator[Request, Response]
```

Execute the authentication flow asynchronously.

By default, this defers to .auth_flow(). You should override this method when the authentication scheme does I/O and/or uses concurrency primitives.

```
async login() \rightarrow None
```

Get a valid OAuth token.

property login_lock: Lock

Make sure that there is a lock in the current event loop.

```
sync_auth_flow(request: Request) → Generator[Request, Response, None]
```

Execute the authentication flow synchronously.

By default, this defers to .auth_flow(). You should override this method when the authentication scheme does I/O and/or uses concurrency primitives.

```
class bimmer_connected.api.authentication.MyBMWLoginClient(*args, **kwargs)
```

Async HTTP client based on httpx. Async Client with automated OAuth token refresh.

```
class bimmer_connected.api.authentication.MyBMWLoginRetry
```

httpx. Auth used as workaround to retry & sleep on 429 Too Many Requests.

```
async async_auth_flow(request: Request) → AsyncGenerator[Request, Response]
```

Execute the authentication flow asynchronously.

By default, this defers to .auth_flow(). You should override this method when the authentication scheme does I/O and/or uses concurrency primitives.

```
sync_auth_flow(request: Request) → Generator[Request, Response, None]
```

Execute the authentication flow synchronously.

By default, this defers to .auth_flow(). You should override this method when the authentication scheme does I/O and/or uses concurrency primitives.

```
bimmer\_connected.api.authentication.get\_retry\_wait\_time(response: Response) \rightarrow int
```

Get the wait time for the next retry from the response and multiply by 2.

8.14.2 bimmer_connected.api.client

```
Generic API management.
```

Async HTTP client based on httpx. Async Client with automated OAuth token refresh.

 $\textbf{generate_default_header}(\textit{brand: CarBrands} \mid \textit{None} = \textit{None}) \rightarrow \textit{Dict}[\textit{str}, \textit{str}]$

Generate a header for HTTP requests to the server.

class bimmer_connected.api.client.MyBMWClientConfiguration(authentication:

MyBMWAuthentication,

log_responses: bool | None = False,
observer_position: GPSPosition | None
= None)

Stores global settings for MyBMWClient.

authentication: MyBMWAuthentication

log_responses: bool | None = False

observer_position: GPSPosition | None = None

 $set_log_responses(log_responses: bool) \rightarrow None$

Set if responses are logged and clear response store.

8.14.3 bimmer_connected.api.regions

Get the right url for the different countries.

bimmer_connected.api.regions.get_app_version(region: Regions) \rightarrow str Get the app version & build number for the region.

bimmer_connected.api.regions.get_ocp_apim_key(region: Regions) \rightarrow str Get the authorization for OAuth settings.

bimmer_connected.api.regions.get_region_from_name(name: str) \rightarrow Regions Get a region for a string.

This function is not case-sensitive.

bimmer_connected.api.regions.get_server_url(region: Regions) \rightarrow str Get the url of the server for the region.

bimmer_connected.api.regions.get_user_agent(region: Regions) \rightarrow str Get the Dart user agent for the region.

 $\label{eq:connected.api.regions.valid_regions()} \rightarrow List[str]$ Get list of valid regions as strings.

8.14.4 bimmer_connected.api.utils

```
Utils for bimmer_connected.api.
```

 $\verb|bimmer_connected.api.utils.anonymize_data| (\textit{json_data: List} \mid \textit{Dict}) \rightarrow \textit{List} \mid \textit{Dict}|$

Replace parts of the logfiles containing personal information.

 $bimmer_connected.api.utils.anonymize_response(response: Response) \rightarrow AnonymizedResponse$

Anonymize a responses URL and content.

bimmer_connected.api.utils.anonymize_vin(match: Match)

Anonymize VINs but keep assignment.

 $bimmer_connected.api.utils.create_s256_code_challenge(code_verifier: str) \rightarrow str$

Create S256 code_challenge with the given code_verifier.

 ${\tt bimmer_connected.api.utils.generate_cn_nonce}(\textit{username: str}) \rightarrow {\tt str}$

Generate a x-login-nonce string.

 $bimmer_connected.api.utils.generate_random_base64_string(size: int) \rightarrow str$

Generate a random base64 string with size.

bimmer_connected.api.utils.generate_token(length: int = 30, chars: str =

'abc defghijk lmnop qr stuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567

._~′) → str

Generate a random token with given length and characters.

bimmer_connected.api.utils.get_capture_position($base64_background_img: str$) \rightarrow str

Get the position of the capture in the background image.

bimmer_connected.api.utils.get_correlation_id() → Dict[str, str]

Generate corrlation headers.

async bimmer_connected.api.utils.handle_httpstatuserror(ex: HTTPStatusError, module: str = 'API',

 $log_handler: Logger \mid None = None,$ $dont_raise: bool = False) \rightarrow None$

Try to extract information from response and re-raise Exception.

bimmer_connected.api.utils.try_import_pillow_image()

Try to import PIL.Image and return if successful.

We only need to load PIL if we are in China, so we try to avoid a general dependency on Pillow for all users. Installing Pillow on Raspberry Pi (ARMv7) is painful.

8.15 bimmer_connected.const

URLs for different services and error code mapping.

Car brands supported by the MyBMW API.

```
BMW = 'bmw'
MINI = 'mini'
```

```
Regions of the world with separate servers.
```

```
CHINA = 'cn'
NORTH_AMERICA = 'na'
REST_OF_WORLD = 'row'
```

8.16 bimmer_connected.models

Generals models used for bimmer_connected.

An anonymized response.

```
content: List | Dict | str | None = None
```

filename: str

class bimmer_connected.models.ChargingSettings($chargingTarget: int \mid None, acLimitValue: int \mid None = None$)

Charging settings to control the vehicle.

```
acLimitValue: int | None = None
```

chargingTarget: int | None

dcLoudness = None

isUnlockCableActive = None

 $\textbf{class} \ \ \textbf{bimmer_connected.models.GPSPosition} (\textit{latitude: float} \mid \textit{None}, \textit{longitude: float} \mid \textit{None})$

GPS coordinates.

latitude: float | None

longitude: float | None

exception bimmer_connected.models.MyBMWAPIError

General BMW API error.

 $\textbf{exception} \ \, \textbf{bimmer_connected.models.MyBMWAuthError}$

Auth-related error from BMW API (HTTP status codes 401 and 403).

exception bimmer_connected.models.MyBMWQuotaError

Quota exceeded on BMW API.

exception bimmer_connected.models.MyBMWRemoteServiceError

Error when executing remote services.

class bimmer_connected.models.**PointOfInterest**(*lat: dataclasses.InitVar[float*], *lon:*

dataclasses.InitVar[float], name: str | None = 'Sent with by bimmer_connected', street: dataclasses.InitVar[str] = None, postal_code: dataclasses.InitVar[str] = None, city: dataclasses.InitVar[str] = None, country: dataclasses.InitVar[str] = None, formattedAddress: str | None = None, address: str | None = None, baseCategoryId: str | None = None, phoneNumber: str | None = None, provider: str | None = None, providerId: str | None = None, providerPoild: str = ", sourceType: str | None = None, type: str | None = None, vehicleCategoryId: str | None = None)

A Point of Interest to be sent to the car. address: str | None = None baseCategoryId: str | None = None city: dataclasses.InitVar[str] = None coordinates: GPSPosition country: dataclasses.InitVar[str] = None entryPoints: List formattedAddress: str | None = None lat: dataclasses.InitVar[float] locationAddress: PointOfInterestAddress | None lon: dataclasses.InitVar[float] name: str | None = 'Sent with by bimmer_connected' phoneNumber: str | None = None postal_code: dataclasses.InitVar[str] = None provider: str | None = None providerId: str | None = None providerPoiId: str = '' sourceType: str | None = None street: dataclasses.InitVar[str] = None

type: str | None = None

vehicleCategoryId: str | None = None

```
class bimmer_connected.models.PointOfInterestAddress(street: str | None = None, postalCode: str |
                                                             None = None, city: str \mid None = None, country:
                                                             str \mid None = None, banchi: str \mid None = None,
                                                             chome: str \mid None = None, countryCode: str \mid
                                                             None = None, district: str | None = None, go:
                                                             str | None = None, houseNumber: str | None =
                                                             None, region: str \mid None = None, regionCode:
                                                             str \mid None = None, settlement: str \mid None =
                                                             None)
     Address data of a PointOfInterest.
     banchi: str | None = None
     chome: str | None = None
     city: str | None = None
     country: str | None = None
     countryCode: str | None = None
     district: str | None = None
     go: str | None = None
     houseNumber: str | None = None
     postalCode: str | None = None
     region: str | None = None
     regionCode: str | None = None
     settlement: str | None = None
     street: str | None = None
class bimmer_connected.models.StrEnum(value, names=None, *, module=None, qualname=None,
                                           type=None, start=1, boundary=None)
     A string enumeration of type (str, Enum). All members are compared via upper(). Defaults to UNKNOWN.
class bimmer_connected.models.ValueWithUnit(value: int | float | None, unit: str | None)
     A value with a corresponding unit.
     unit: str | None
          Alias for field number 1
     value: int | float | None
          Alias for field number 0
class bimmer_connected.models.VehicleDataBase
     A base class for parsing and storing complex vehicle data.
     classmethod from_vehicle_data(vehicle data: Dict)
          Create the class based on vehicle data from API.
     update_from_vehicle_data(vehicle data: Dict)
```

Update the attributes based on vehicle data from API.

8.17 bimmer_connected.utils

General utils and base classes used in the library.

JSON Encoder that handles data classes, properties and additional data types.

```
default(o) \rightarrow str | dict
```

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

bimmer_connected.utils.get_class_property_names(obj: object)

Return the names of all properties of a class.

```
bimmer_connected.utils.log_response_store_to_file(response_store: List[AnonymizedResponse], logfile\ path:\ Path) \rightarrow None
```

Log all responses to files.

```
\verb|bimmer_connected.utils.parse_datetime| (\textit{date\_str: str}) \rightarrow \textit{datetime} \mid \textit{None}
```

Convert a time string into datetime.

```
bimmer_connected.utils.to_camel_case(input_str: str) \rightarrow str Convert SNAKE_CASE or snake_case to camelCase.
```

8.18 bimmer_connected.vehicle

The bimmer_connected.vehicle module contains all data & parsers for a vehicle.

8.18.1 bimmer_connected.vehicle.vehicle

Models state and remote services of one vehicle.

Known Values for lsc_type field.

Not really sure, what this value really contains.

```
ACTIVATED = 'ACTIVATED'

NOT_CAPABLE = 'NOT_CAPABLE'

NOT_SUPPORTED = 'NOT_SUPPORTED'

UNKNOWN = 'UNKNOWN'
```

Models state and remote services of one vehicle.

Parameters

- account MyBMW account this vehicle belongs to
- attributes attributes of the vehicle as provided by the server

property available_attributes: List[str]

Get the list of non-drivetrain attributes available for this vehicle.

property brand: CarBrands

Get the car brand.

 $combine_data(data: Dict | List[Dict], fetched_at: datetime | None = None) \rightarrow Dict$

Combine API responses and additional information to a single dictionary.

property drive_train: DriveTrainType

Get the type of drive train of the vehicle.

property drive_train_attributes: List[str]

Get list of attributes available for the drive train of the vehicle.

The list of available attributes depends if on the type of drive train. Some attributes only exist for electric/hybrid vehicles, others only if you have a combustion engine. Depending on the state of the vehicle, some of the attributes might still be None.

async get_vehicle_image(direction: VehicleViewDirection) \rightarrow bytes

Get a rendered image of the vehicle.

:returns bytes containing the image in PNG format.

async get_vehicle_state() \rightarrow None

Retrieve vehicle data from BMW servers.

property has_combustion_drivetrain: bool

Return True if vehicle is equipped with an internal combustion engine.

In this case we can get the state of the gas tank.

property has_electric_drivetrain: bool

Return True if vehicle is equipped with a high voltage battery.

In this case we can get the state of the battery in the state attributes.

property is_charging_plan_supported: bool

Return True if charging profile is available and can be set via API.

property is_charging_settings_supported: bool

Return True if charging settings can be set via API.

property is_lsc_enabled: bool

Return True if LastStateCall is enabled (vehicle automatically updates API).

property is_remote_charge_start_enabled: bool

Return True if charging can be started via the API.

property is_remote_charge_stop_enabled: bool

Return True if charging can be stop via the API.

property is_remote_climate_start_enabled: bool

Return True if AC/ventilation can be started via the API.

property is_remote_climate_stop_enabled: bool

Return True if AC/ventilation can be stopped via the API.

property is_remote_horn_enabled: bool

Return True if the horn can be activated via the API.

property is_remote_lights_enabled: bool

Return True if the lights can be activated via the API.

property is_remote_lock_enabled: bool

Return True if vehicle can be locked via the API.

property is_remote_sendpoi_enabled: bool

Return True if POIs can be set via the API.

property is_remote_set_ac_limit_enabled: bool

Return True if AC limit can be set via the API.

property is_remote_set_target_soc_enabled: bool

Return True if Target SoC can be set via the API.

property is_remote_unlock_enabled: bool

Return True if POIs can be unlocked via the API.

property is_vehicle_active: bool

Deprecated, always returns False.

Check if the vehicle is active/moving.

If the vehicle was active/moving at the time of the last status update, current position is not available.

property is_vehicle_tracking_enabled: bool

Return True if vehicle finder is enabled in vehicle.

property lsc_type: LscType

Get the lscType of the vehicle.

Not really sure what that value really means. If it is NOT_CAPABLE, that probably means that the vehicle state will not contain much data.

property mileage: ValueWithUnit

Get the mileage of the vehicle.

property name: str

Get the name of the vehicle.

```
property timestamp: datetime | None
```

Get the timestamp when the data was recorded.

 $\textbf{update_state}(\textit{data: Dict} \mid \textit{List[Dict]}, \textit{fetched_at: datetime} \mid \textit{None} = \textit{None}) \rightarrow \textit{None}$

Update the state of a vehicle.

```
property vin: str
```

Get the VIN (vehicle identification number) of the vehicle.

 $\begin{tabular}{ll} \textbf{class} & \textbf{bimmer_connected.vehicle.vehicle.VehicleViewDirection} (value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}$

Viewing angles for the vehicle.

This is used to get a rendered image of the vehicle.

```
FRONT = 'FrontView'
FRONTSIDE = 'AngleSideViewForty'
SIDE = 'SideViewLeft'
UNKNOWN = 'UNKNOWN'
```

8.18.2 bimmer_connected.vehicle.remote_services

Trigger remote services on a vehicle.

Enumeration of possible states of the execution of a remote service.

```
DELIVERED = 'DELIVERED'

ERROR = 'ERROR'

EXECUTED = 'EXECUTED'

IGNORED = 'IGNORED'

INITIATED = 'INITIATED'

PENDING = 'PENDING'

UNKNOWN = 'UNKNOWN'
```

Wraps the status of the execution of a remote service.

 $\textbf{class} \ \texttt{bimmer_connected.vehicle.remote_services.} \textbf{RemoteServices} (\textit{vehicle}: \ \texttt{MyBMWVehicle})$

Trigger remote services on a vehicle.

```
async trigger_charge_start() → RemoteServiceStatus
```

Trigger the vehicle to start charging.

```
async trigger_charge_stop() \rightarrow RemoteServiceStatus
           Trigger the vehicle to stop charging.
     async trigger_charging_profile_update(charging_mode: ChargingMode | None = None,
                                                    precondition climate: bool | None = None) \rightarrow
                                                    RemoteServiceStatus
           Update the charging profile on the vehicle.
     async trigger_charging_settings_update(target_soc: int | None = None, ac_limit: int | None = None)
                                                     \rightarrow RemoteServiceStatus
           Update the charging settings on the vehicle.
     async trigger_remote_air_conditioning() \rightarrow RemoteServiceStatus
           Trigger the air conditioning to start.
     async trigger_remote_air_conditioning_stop() \rightarrow RemoteServiceStatus
           Trigger the air conditioning to stop.
     async trigger_remote_door_lock() \rightarrow RemoteServiceStatus
           Trigger the vehicle to lock its doors.
     async trigger_remote_door_unlock() → RemoteServiceStatus
           Trigger the vehicle to unlock its doors.
     async trigger_remote_horn() → RemoteServiceStatus
           Trigger the vehicle to sound its horn.
     async trigger_remote_light_flash() → RemoteServiceStatus
           Trigger the vehicle to flash its headlights.
     async trigger_remote_service(service_id: Services, params: Dict | None = None, data: Any = None,
                                         refresh: bool = False) \rightarrow RemoteServiceStatus
           Trigger a remote service and wait for the result.
     async trigger_remote_vehicle_finder() \rightarrow RemoteServiceStatus
           Trigger the vehicle finder.
     async trigger_send_poi(poi: PointOfInterest | Dict) \rightarrow RemoteServiceStatus
           Send a PointOfInterest to the vehicle.
               Parameters
                   poi - A PointOfInterest containing at least 'lat' and 'lon' and optionally 'name', 'street',
                   'city', 'postalCode', 'country'
class bimmer_connected.vehicle.remote_services.Services(value, names=None, *, module=None,
                                                                     qualname=None, type=None, start=1,
                                                                     boundary=None)
     Enumeration of possible services to be executed.
     AIR_CONDITIONING = 'climate-now'
     CHARGE_START = 'start-charging'
     CHARGE_STOP = 'stop-charging'
     CHARGING_PROFILE = 'CHARGING_PROFILE'
     CHARGING_SETTINGS = 'CHARGING_SETTINGS'
```

```
DOOR_LOCK = 'door-lock'
DOOR_UNLOCK = 'door-unlock'
HORN = 'horn-blow'
LIGHT_FLASH = 'light-flash'
SEND_POI = 'SEND_POI'
VEHICLE_FINDER = 'vehicle-finder'
```

8.18.3 bimmer_connected.vehicle.charging_profile

Models the charging profiles of a vehicle.

```
\begin{tabular}{ll} {\bf class \ bimmer\_connected.vehicle.charging\_profile.ChargingMode}(value, names=None, *, \\ module=None, qualname=None, \\ type=None, start=1, \\ boundary=None) \end{tabular}
```

Charging mode of electric vehicle.

```
DELAYED_CHARGING = 'DELAYED_CHARGING'

IMMEDIATE_CHARGING = 'IMMEDIATE_CHARGING'

UNKNOWN = 'UNKNOWN'
```

class bimmer_connected.vehicle.charging_profile.ChargingPreferences(value, names=None, *,

module=None, qualname=None, type=None, start=1, boundary=None)

Charging preferences of electric vehicle.

```
CHARGING_WINDOW = 'CHARGING_WINDOW'

NO_PRESELECTION = 'NO_PRESELECTION'

UNKNOWN = 'UNKNOWN'
```

class bimmer_connected.vehicle.charging_profile.ChargingProfile(is_pre_entry_climatization_enabled:

```
bool, timer_type: TimerTypes,
departure_times:
List[DepartureTimer],
preferred_charging_window:
ChargingWindow,
charging_preferences:
ChargingPreferences,
charging_mode: ChargingMode,
ac_current_limit: int | None =
None, ac_available_limits: list |
None = None, charg-
ing_preferences_service_pack:
str | None = None)
```

Models the charging profile of a vehicle.

```
ac_available_limits: list | None = None
          Available AC limits to be selected.
     ac_current_limit: int | None = None
          Returns the ac current limit.
     charging_mode: ChargingMode
          Returns the preferred charging mode.
     charging_preferences: ChargingPreferences
          Returns the preferred charging preferences.
     charging_preferences_service_pack: str | None = None
          Service Pack required for remote service format.
     departure_times: List[DepartureTimer]
          List of timers.
     format\_for\_remote\_service() \rightarrow dict
          Format current charging profile as base to be sent to remote service.
     is_pre_entry_climatization_enabled: bool
          Get status of pre-entry climatization.
     preferred_charging_window: ChargingWindow
          Returns the preferred charging window.
     timer_type: TimerTypes
          Returns the current timer plan type.
class bimmer_connected.vehicle.charging_profile.ChargingWindow(window_dict: dict)
     A charging window.
     property end_time: time
          End of the charging window.
     property start_time: time
          Start of the charging window.
class bimmer_connected.vehicle.charging_profile.DepartureTimer(timer_dict: dict)
     A departure timer.
     property action: str | None
          What does the timer do.
     property start_time: time | None
          Deperture time for this timer.
     property timer_id: int | None
          ID of this timer.
     property weekdays: List[str]
          Active weekdays for this timer.
class bimmer_connected.vehicle.charging_profile.TimerTypes(value, names=None, *, module=None,
                                                                   qualname=None, type=None, start=1,
                                                                   boundary=None)
     Different timer types.
```

```
TWO_TIMES_TIMER = 'TWO_TIMES_TIMER'
UNKNOWN = 'UNKNOWN'
WEEKLY_PLANNER = 'WEEKLY_PLANNER'
```

8.18.4 bimmer_connected.vehicle.doors_windows

```
Models the state of a vehicle.
```

```
class bimmer_connected.vehicle.doors_windows.DoorsAndWindows(door_lock_state: ~bim-
                                                                       mer_connected.vehicle.doors_windows.LockState
                                                                       = LockState.UNKNOWN, lids: ~typ-
                                                                       ing.List[~bimmer_connected.vehicle.doors_windows.Li
                                                                       = <factory>, windows: ~typ-
                                                                       ing.List[~bimmer_connected.vehicle.doors_windows.W
                                                                       = < factory > )
     Provides an accessible version of properties.doorsAndWindows.
     property all_lids_closed: bool
          Check if all lids are closed.
     property all_windows_closed: bool
          Check if all windows are closed.
```

door lock state: LockState = 'UNKNOWN'

Get state of the door locks.

lids: List[Lid]

All lids (doors+hood+trunk) of the car.

property open_lids: List[Lid]

Get all open lids of the car.

property open_windows: List[Window]

Get all open windows of the car.

windows: List[Window]

All windows (doors+sunroof) of the car.

class bimmer_connected.vehicle.doors_windows.**Lid**(name: str, state: str)

A lid of the vehicle.

Lids are: Doors + Trunk + Hatch property is_closed: bool Check if the lid is closed.

name

name of the lid

class bimmer_connected.vehicle.doors_windows.**LidState**(value, names=None, *, module=None, qualname=None, type=None, start=1,boundary=None)

Possible states of the hatch, trunk, doors, windows, sun roof.

```
CLOSED = 'CLOSED'
     INTERMEDIATE = 'INTERMEDIATE'
     INVALID = 'INVALID'
     OPEN = 'OPEN'
     OPEN_TILT = 'OPEN_TILT'
     UNKNOWN = 'UNKNOWN'
class bimmer_connected.vehicle.doors_windows.LockState(value, names=None, *, module=None,
                                                           qualname=None, type=None, start=1,
                                                           boundary=None)
     Possible states of the door locks.
     LOCKED = 'LOCKED'
     PARTIALLY_LOCKED = 'PARTIALLY_LOCKED'
     SECURED = 'SECURED'
     SELECTIVE_LOCKED = 'SELECTIVE_LOCKED'
     UNKNOWN = 'UNKNOWN'
     UNLOCKED = 'UNLOCKED'
class bimmer_connected.vehicle.doors_windows.Window(name: str, state: str)
     A window of the vehicle.
     A window can be a normal window of the car or the sun roof.
8.18.5 bimmer_connected.vehicle.fuel_and_battery
Generals models used for bimmer_connected.
class bimmer_connected.vehicle.fuel_and_battery.ChargingState(value, names=None, *,
                                                                   module=None, qualname=None,
                                                                   type=None, start=1,
                                                                   boundary=None)
     Charging state of electric vehicle.
     CHARGING = 'CHARGING'
     COMPLETE = 'COMPLETE'
     DEFAULT = 'DEFAULT'
     ERROR = 'ERROR'
     FINISHED_FULLY_CHARGED = 'FINISHED_FULLY_CHARGED'
     FINISHED_NOT_FULL = 'FINISHED_NOT_FULL'
     FULLY_CHARGED = 'FULLY_CHARGED'
```

```
NOT_CHARGING = 'NOT_CHARGING'
     PLUGGED_IN = 'PLUGGED_IN'
     TARGET_REACHED = 'TARGET_REACHED'
     UNKNOWN = 'UNKNOWN'
     WAITING_FOR_CHARGING = 'WAITING_FOR_CHARGING'
class bimmer_connected.vehicle.fuel_and_battery.FuelAndBattery(remaining_range_fuel:
                                                                          ValueWithUnit \mid None = (None,
                                                                          None), remaining_range_electric:
                                                                          ValueWithUnit \mid None = (None,
                                                                          None), remaining_range_total:
                                                                          ValueWithUnit \mid None = (None,
                                                                          None), remaining_fuel:
                                                                          ValueWithUnit \mid None = (None,
                                                                          None), remaining_fuel_percent:
                                                                          int \mid None = None,
                                                                          remaining_battery_percent: int |
                                                                          None = None, charging\_status:
                                                                          ChargingState | None = None,
                                                                          charging_start_time: datetime |
                                                                          None = None, charging end time:
                                                                          datetime \mid None = None,
                                                                          is_charger_connected: bool =
                                                                          False, charging_target: int | None
                                                                          = None)
     Provides an accessible version of status. Fuel And Battery.
     charging_end_time: datetime | None = None
          The estimated time the vehicle will have finished charging.
     charging_start_time: datetime | None = None
          The planned time the vehicle will start charging in UTC.
     charging_status: ChargingState | None = None
          Charging state of the vehicle.
     charging_target: int | None = None
          State of charging target in percent.
     classmethod from_vehicle_data(vehicle data: Dict)
          Create the class based on vehicle data from API.
     is charger connected: bool = False
          Get status of the connection
     remaining_battery_percent: int | None = None
          State of charge of the high voltage battery in percent.
     remaining_fuel: ValueWithUnit | None = (None, None)
          Get the remaining fuel of the vehicle.
```

INVALID = 'INVALID'

```
remaining_fuel_percent: int | None = None
    State of charge of the high voltage battery in percent.
remaining_range_electric: ValueWithUnit | None = (None, None)
    Get the remaining range of the vehicle on electricity.
remaining_range_fuel: ValueWithUnit | None = (None, None)
    Get the remaining range of the vehicle on fuel.
remaining_range_total: ValueWithUnit | None = (None, None)
    Get the total remaining range of the vehicle (fuel + electricity, if available).
```

8.18.6 bimmer_connected.vehicle.location

Generals models used for bimmer_connected.

```
class bimmer_connected.vehicle.location.VehicleLocation(location: GPSPosition | None = None,
                                                                  heading: int \mid None = None,
                                                                  vehicle_update_timestamp: datetime | None
                                                                  = None, account_region: Regions | None =
                                                                  None, remote service position: Dict
                                                                  None = None)
```

```
The current position of a vehicle.
account_region: Regions | None = None
classmethod from_vehicle_data(vehicle data: Dict)
    Create the class based on vehicle data from API.
heading: int | None = None
    The last known heading/direction of the vehicle.
location: GPSPosition | None = None
    The last known position of the vehicle.
remote_service_position: Dict | None = None
set_remote_service_position(remote_service_dict: Dict)
```

Store remote service position returned from vehicle finder service.

vehicle_update_timestamp: datetime | None = None

8.18.7 bimmer_connected.vehicle.reports

Models the state of a vehicle.

```
class bimmer_connected.vehicle.reports.CheckControlMessage(description_short: str,
                                                                    description_long: str | None, state:
                                                                    CheckControlStatus)
```

Check control message sent from the server.

```
description_long: str | None
description_short: str
```

```
classmethod from_api_entry(type: str, severity: str, longDescription: str | None = None, **kwargs)
          Parse a check control entry from the API format to CheckControlMessage.
     state: CheckControlStatus
class bimmer_connected.vehicle.reports.CheckControlMessageReport(messages: ~typ-
                                                                           ing.List[~bimmer connected.vehicle.reports.Chec
                                                                           = < factory>,
                                                                           has_check_control_messages:
                                                                           bool = False)
     Parse and summarizes check control messages (e.g. low tire pressure).
     has_check_control_messages: bool = False
          Indicate if check control messages are present.
     messages: List[CheckControlMessage]
          List of check control messages.
class bimmer_connected.vehicle.reports.CheckControlStatus(value, names=None, *, module=None,
                                                                   qualname=None, type=None, start=1,
                                                                   boundary=None)
     Status of the condition based services.
     CRITICAL = 'CRITICAL'
     HIGH = 'HIGH'
     LOW = 'LOW'
     MEDIUM = 'MEDIUM'
     OK = 'OK'
     UNKNOWN = 'UNKNOWN'
class bimmer_connected.vehicle.reports.ConditionBasedService(service_type: str, state:
                                                                      ConditionBasedServiceStatus,
                                                                      due_date: datetime | None,
                                                                      due_distance: ValueWithUnit)
     Entry in the list of condition based services.
     due_date: datetime | None
     due_distance: ValueWithUnit
     classmethod from_api_entry(type: str, status: str, dateTime: str | None = None, mileage: int | None =
                                    None, **kwargs)
          Parse a condition based service entry from the API format to ConditionBasedService.
     service_type: str
     state: ConditionBasedServiceStatus
class bimmer_connected.vehicle.reports.ConditionBasedServiceReport(messages: ~typ-
                                                                             ing.List[~bimmer_connected.vehicle.reports.Co
                                                                             = < factory>,
                                                                             is_service_required: bool =
                                                                             False)
```

```
Parse and summarizes condition based services (e.g. next oil service).
     is_service_required: bool = False
          Indicate if a service is required.
     messages: List[ConditionBasedService]
          List of the condition based services.
class bimmer_connected.vehicle.reports.ConditionBasedServiceStatus(value, names=None, *,
                                                                             module=None,
                                                                             qualname=None,
                                                                             type=None, start=1,
                                                                             boundary=None)
     Status of the condition based services.
     OK = 'OK'
     OVERDUE = 'OVERDUE'
     PENDING = 'PENDING'
     UNKNOWN = 'UNKNOWN'
class bimmer_connected.vehicle.reports.Headunit(idrive_version: str = ", headunit_type: str = ",
                                                       software\_version: str = ")
     Parse and summarizes headunit hard/software versions.
     headunit_type: str = ''
          Type of headunit.
     idrive_version: str = ''
          IDRIVE generation.
     software_version: str = ''
          Current software revision of vehicle
```

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

```
bimmer_connected.account, 30
bimmer_connected.api.authentication, 32
bimmer_connected.api.client, 33
bimmer_connected.api.regions, 33
bimmer_connected.api.utils, 34
bimmer_connected.const, 34
bimmer_connected.models, 35
bimmer_connected.utils, 38
bimmer_connected.vehicle.charging_profile, 43
bimmer_connected.vehicle.doors_windows, 45
bimmer_connected.vehicle.fuel_and_battery, 46
bimmer_connected.vehicle.location, 48
bimmer_connected.vehicle.remote_services, 41
bimmer_connected.vehicle.reports, 48
bimmer_connected.vehicle.vehicle, 38
```

bimmer conn	ected	Documer	ntation
-------------	-------	---------	---------

54 Python Module Index

INDEX

A	method), 32
ac_available_limits (bim-	authentication (bim-
mer_connected.vehicle.charging_profile.Charging	gProfile mer_connected.api.client.MyBMWClientConfiguration attribute), 33
attribute), 43	available_attributes (bim-
ac_current_limit (bim-	
mer_connected.vehicle.charging_profile.Charging attribute), 44	property), 39
account_region (bim-	В
$mer_connected.vehicle.location.VehicleLocation$	
attribute), 48	banchi (bimmer_connected.models.PointOfInterestAddress
$\verb"acLimitValue" (bimmer_connected.models. Charging Setting S$	gs attribute), 37
attribute), 35	baseCategoryId (bim-
<pre>action(bimmer_connected.vehicle.charging_profile.Depair</pre>	rtureTimermer_connected.models.PointOfInterest at- tribute), 36
ACTIVATED (bimmer_connected.vehicle.vehicle.LscType	bimmer_connected.account
attribute), 38	module, 30
<pre>add_vehicle() (bimmer_connected.account.MyBMWAccount.</pre>	himmer_connected.api.authentication
method), 30	module, 32
<pre>address (bimmer_connected.models.PointOfInterest at-</pre>	bimmer_connected.api.client
tribute), 36	module, 33
AIR_CONDITIONING (bim-	bimmer_connected.api.regions
mer_connected.vehicle.remote_services.Services	module, 33
attribute), 42	bimmer_connected.api.utils
all_lids_closed (bim-	module, 34
mer_connected.vehicle.doors_windows.DoorsAnd	/Minnlewsconnected.const
property), 45	module, 34
all_windows_closed (bim-	bimmer_connected.models
mer_connected.vehicle.doors_windows.DoorsAnd	lWindmodule, 35
property), 45	bimmer_connected.utils
anonymize_data() (in module bim-	module, 38
mer_connected.api.utils), 34	bimmer_connected.vehicle.charging_profile
anonymize_response() (in module bim-	module, 43
mer_connected.api.utils), 34	bimmer_connected.vehicle.doors_windows
anonymize_vin() (in module bim-	module, 45
mer_connected.api.utils), 34	bimmer_connected.vehicle.fuel_and_battery
AnonymizedResponse (class in bim-	module, 46
mer_connected.models), 35	bimmer_connected.vehicle.location
async_auth_flow() (bim-	module, 48
*	himmen_connected.vehicle.remote_services
method), 32	module, 41
•	bimmer_connected.vehicle.reports
$mer_connected.api.authentication.MyBMWLogin$	Retrymodule, 48

bimmer_connected.vehicle.vehicle module, 38		$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
BMW (bimmer_connected.const.CarBrands attribute), brand (bimmer_connected.vehicle.vehicle.MyBMWV property), 39		mer_connected.models.ChargingSettings	
С		attribute), 35 ChargingWindow (class in bim-	
CarBrands (class in bimmer_connected.const), 34		mer_connected.vehicle.charging_profile),	
CHARGE_START (bimmer_connected.vehicle.remote_s attribute), 42	services	s.Services 44 CheckControlMessage (class in bim-	
CHARGE_STOP (bimmer_connected.vehicle.remote_se attribute), 42	rvices.	Services mer_connected.vehicle.reports), 48 CheckControlMessageReport (class in bim-	
attribute), 42 CHARGING (bimmer_connected.vehicle.fuel_and_batt			
attribute), 46	cr y. Cru	CheckControlStatus (class in bim-	
charging end time (bim-	mer_connected.vehicle.reports), 49	
mer_connected.vehicle.fuel_and_battery.Fi attribute), 47	uelAnd	Chome (bimmer_connected.const.Regions attribute), 35 chome (bimmer_connected.models.PointOfInterestAddress	
charging_mode(bimmer_connected.vehicle.chargin attribute), 44	ig_prof	file.Chargiff t^{pi}t offte), 37 city (bimmer_connected.models.PointOfInterest at-	
charging preferences (bim-	tribute), 36	
mer_connected.vehicle.charging_profile.Cl attribute), 44	harging	g qi,ty ll(bimmer_connected.models.PointOfInterestAddress attribute), 37	
	bim-	${\tt CLOSED}(bimmer_connected.vehicle.doors_windows.LidState$	
mer_connected.vehicle.charging_profile.Cl attribute), 44	harging	gProfile attribute), 45 combine_data() (bim-	
CHARGING_PROFILE (bim-	mer_connected.vehicle.vehicle.MyBMWVehicle	
mer_connected.vehicle.remote_services.Set		method), 39	
attribute), 42		COMPLETE (bimmer_connected.vehicle.fuel_and_battery.ChargingS	tate
CHARGING_SETTINGS (bim-	attribute), 46 ConditionBasedService (class in bim-	
mer_connected.vehicle.remote_services.Ser attribute), 42		mer_connected.vehicle.reports), 49	
charging_start_time (bim-	ConditionBasedServiceReport (class in bim-	
mer_connected.vehicle.fuel_and_battery.Fi attribute), 47	uelAnd	ConditionBasedServiceStatus (class in bim-	
charging_status (bim-	mer_connected.vehicle.reports), 50	
<pre>mer_connected.vehicle.fuel_and_battery.Fi attribute), 47</pre>		ggnfjg (bimmer_connected.account.MyBMWAccount at- tribute), 31	
charging_target (bim-	$\verb"content" (bimmer_connected.models. A nonymized Response$	
mer_connected.vehicle.fuel_and_battery.Fu	uelAnd	Battery attribute), 35	
attribute), 47		coordinates (bimmer_connected.models.PointOfInterest	
mer_connected.vehicle.charging_profile.Cl	bim- harging	attribute), 36 gqquptxycekimmer_connected.models.PointOfInterest_at-	
attribute), 43		tribute), 36	
mer_connected.vehicle.charging_profile),		country (bimmer_connected.models.PointOfInterestAddress attribute), 37	
43		countryCode (bimmer_connected.models.PointOfInterestAddress	
mer_connected.vehicle.charging_profile),	bim-	attribute), 37 create_s256_code_challenge() (in module bim-	
43	ı ·	mer_connected.api.utils), 34 CRITICAL (bimmer_connected.vehicle.reports.CheckControlStatus	
mer_connected.vehicle.charging_profile),	bim-	attribute), 49	
43 CharaingSottings (class in	bim-	D	
ChargingSettings (class in mer_connected.models), 35	vim-	dcLoudness (bimmer_connected.models.ChargingSettings	

```
attribute), 35
                                                      EXECUTED (bimmer connected.vehicle.remote services.ExecutionState
DEFAULT (bimmer connected vehicle fuel and battery. Charging State attribute), 41
         attribute), 46
                                                      ExecutionState
                                                                               (class
                                                                                            in
                                                                                                      bim-
default() (bimmer_connected.utils.MyBMWJSONEncoder
                                                               mer_connected.vehicle.remote_services),
         method), 38
DELAYED_CHARGING
                                               (bim-
        mer_connected.vehicle.charging_profile.ChargingMode
        attribute), 43
                                                      filename (bimmer connected.models.AnonymizedResponse
DELIVERED (bimmer_connected.vehicle.remote_services.ExecutionStatetribute), 35
         attribute), 41
                                                      FINISHED_FULLY_CHARGED
departure_times
                                               (bim-
                                                               mer connected.vehicle.fuel and battery.ChargingState
        mer\_connected.vehicle.charging\_profile.ChargingProfile
                                                               attribute), 46
        attribute), 44
                                                      FINISHED_NOT_FULL
                                                                                                      (bim-
DepartureTimer
                        (class
                                                bim-
                                                               mer_connected.vehicle.fuel_and_battery.ChargingState
                                      in
         mer_connected.vehicle.charging_profile),
                                                               attribute), 46
         44
                                                      format_for_remote_service()
                                                                                                      (bim-
description_long
                                               (bim-
                                                               mer_connected.vehicle.charging_profile.ChargingProfile
        mer connected.vehicle.reports.CheckControlMessage
                                                               method), 44
        attribute), 48
                                                      formattedAddress
                                                                                                      (bim-
description_short
                                               (bim-
                                                               mer_connected.models.PointOfInterest
                                                                                                        at-
        mer_connected.vehicle.reports.CheckControlMessage
                                                               tribute), 36
        attribute), 48
                                                      from_api_entry()
                                                                                                      (bim-
{\tt district}\ (bimmer\_connected.models.PointOfInterestAddress
                                                               mer connected.vehicle.reports.CheckControlMessage
         attribute), 37
                                                               class method), 48
DOOR_LOCK(bimmer_connected.vehicle.remote_services.Serfricons_api_entry()
                                                                                                      (bim-
         attribute), 42
                                                               mer_connected.vehicle.reports.ConditionBasedService
door_lock_state
                                               (bim-
                                                               class method), 49
        mer_connected.vehicle.doors_windows.DoorsAnd \( \frac{Mindowe}{1} \) indowehicle_data()
                                                                                                      (bim-
        attribute), 45
                                                               mer connected.models.VehicleDataBase
DOOR_UNLOCK (bimmer_connected.vehicle.remote_services.Services class method), 37
         attribute), 43
                                                      from_vehicle_data()
DoorsAndWindows
                         (class
                                      in
                                                bim-
                                                               mer_connected.vehicle.fuel_and_battery.FuelAndBattery
        mer_connected.vehicle.doors_windows),
                                                               class method), 47
                                                      from_vehicle_data()
                                                                                                      (bim-
drive_train(bimmer connected.vehicle.vehicle.MvBMWVehicle
                                                               mer_connected.vehicle.location.VehicleLocation
        property), 39
                                                               class method), 48
drive_train_attributes
                                               (bim-
                                                      {\tt FRONT}\,(bimmer\_connected.vehicle.Vehicle.VehicleViewDirection
        mer_connected.vehicle.vehicle.MyBMWVehicle
                                                               attribute), 41
        property), 39
                                                      FRONTSIDE (bimmer connected.vehicle.vehicle.VehicleViewDirection
due_date(bimmer_connected.vehicle.reports.ConditionBasedServiceattribute), 41
         attribute), 49
                                                      FuelAndBatterv
                                                                               (class
                                                                                                      bim-
due_distance(bimmer_connected.vehicle.reports.ConditionBasedSaneiceconnected.vehicle.fuel and battery),
         attribute), 49
                                                      FULLY_CHARGED (bimmer connected.vehicle.fuel and battery.ChargingSta
F
                                                               attribute), 46
\verb"end_time" (bimmer\_connected.vehicle.charging\_profile.ChargingWindow) \\
        property), 44
attribute), 36
                                                               erty), 31
ERROR (bimmer_connected.vehicle.fuel_and_battery.Chargingshiete_te_te_n_nonce()
                                                                                          module
                                                                                                      bim-
         attribute), 46
                                                               mer_connected.api.utils), 34
ERROR (bimmer_connected.vehicle.remote_services.Executionstate_default_header()
                                                                                                      (bim-
         attribute), 41
                                                               mer_connected.api.client.MyBMWClient
```

method), 33

<pre>generate_random_base64_string() (in module bim- mer_connected.api.utils), 34</pre>	Headunit (class in bimmer_connected.vehicle.reports), 50
<pre>generate_token() (in module bim- mer_connected.api.utils), 34</pre>	headunit_type (bimmer_connected.vehicle.reports.Headunit attribute), 50
<pre>get_app_version() (in module bim-</pre>	HIGH (bimmer_connected.vehicle.reports.CheckControlStatus
mer_connected.api.regions), 33	attribute), 49
<pre>get_capture_position() (in module bim-</pre>	${\tt HORN}(bimmer_connected.vehicle.remote_services.Services$
mer_connected.api.utils), 34	attribute), 43
<pre>get_class_property_names() (in module bim- mer_connected.utils), 38</pre>	houseNumber (bimmer_connected.models.PointOfInterestAddress attribute), 37
<pre>get_correlation_id() (in module bim- mer_connected.api.utils), 34</pre>	I
<pre>get_ocp_apim_key() (in module bim-</pre>	idrive_version (bim-
mer_connected.api.regions), 33	mer_connected.vehicle.reports.Headunit
<pre>get_region_from_name() (in module bim-</pre>	attribute), 50
mer_connected.api.regions), 33	${\tt IGNORED}\ (bimmer_connected. vehicle. remote_services. Execution State$
<pre>get_retry_wait_time() (in module bim-</pre>	attribute), 41
mer_connected.api.authentication), 32	IMMEDIATE_CHARGING (bim-
<pre>get_server_url() (in module bim-</pre>	mer_connected.vehicle.charging_profile.ChargingMode
mer_connected.api.regions), 33	attribute), 43
<pre>get_stored_responses()</pre>	${\tt INITIATED} (bimmer_connected. vehicle. remote_services. Execution State$
mer_connected.account.MyBMWAccount	attribute), 41
static method), 31	INTERMEDIATE (bimmer_connected.vehicle.doors_windows.LidState
get_user_agent() (in module bim-	attribute), 46
mer_connected.api.regions), 33	INVALID (bimmer_connected.vehicle.doors_windows.LidState
<pre>get_vehicle() (bimmer_connected.account.MyBMWAccount</pre>	,
get_vehicle_image() (bim-	INVALID (bimmer_connected.vehicle.fuel_and_battery.ChargingState
mer_connected.vehicle.vehicle.MyBMWVehicle	attribute), 46
method), 39	is_charger_connected (bim-
get_vehicle_state() (bim-	mer_connected.vehicle.fuel_and_battery.FuelAndBattery
mer_connected.vehicle.vehicle.MyBMWVehicle	attribute), 47 is_charging_plan_supported (bim-
method), 39	<pre>is_charging_plan_supported</pre>
get_vehicles() (bim-	property), 39
mer_connected.account.MyBMWAccount	is_charging_settings_supported (bim-
method), 31	mer_connected.vehicle.vehicle.MyBMWVehicle
go (bimmer_connected.models.PointOfInterestAddress	property), 39
attribute), 37	is_closed(bimmer_connected.vehicle.doors_windows.Lid
GPSPosition (class in bimmer_connected.models), 35	property), 45
	is_lsc_enabled (bim-
H	mer_connected.vehicle.vehicle.MyBMWVehicle
handle_httpstatuserror() (in module bim-	property), 39
mer_connected.api.utils), 34	is_pre_entry_climatization_enabled (bim-
has_check_control_messages (bim-	mer_connected.vehicle.charging_profile.ChargingProfile
$mer_connected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.reports.CheckControlMected.vehicle.vehic$	ssageReportuttribute), 44
attribute), 49	is_remote_charge_start_enabled (bim-
has_combustion_drivetrain (bim-	mer_connected.vehicle.vehicle.MyBMWVehicle
mer_connected.vehicle.vehicle.MyBMWVehicle	property), 40
property), 39	is_remote_charge_stop_enabled (bim-
has_electric_drivetrain (bim-	mer_connected.vehicle.vehicle.MyBMWVehicle
mer_connected.vehicle.vehicle.MyBMWVehicle	property), 40
property), 39	is_remote_climate_start_enabled (bim-
heading (bimmer_connected.vehicle.location.VehicleLocation.yehicleLocation). 48	ation mer_connected.vehicle.vehicle.MyBMWVehicle property), 40
ин илиет. 40	DIODELLY), 4U

is_remote_climate_stop_enabled (bim-	locationAddress (bim-
mer_connected.vehicle.vehicle.MyBMWVehicle property), 40	mer_connected.models.PointOfInterest at- tribute), 36
is_remote_horn_enabled (bim-	LOCKED (bimmer_connected.vehicle.doors_windows.LockState
mer_connected.vehicle.vehicle.MyBMWVehicle	attribute), 46
property), 40	LockState (class in bim-
is_remote_lights_enabled (bim-	mer_connected.vehicle.doors_windows),
mer_connected.vehicle.vehicle.MyBMWVehicle	46
property), 40	<pre>log_response_store_to_file() (in module bim-</pre>
is_remote_lock_enabled (bim-	mer_connected.utils), 38
mer_connected.vehicle.vehicle.MyBMWVehicle property), 40	log_responses (bimmer_connected.account.MyBMWAccount attribute), 31
is_remote_sendpoi_enabled (bim-	log_responses(bimmer_connected.api.client.MyBMWClientConfiguration)
mer_connected.vehicle.vehicle.MyBMWVehicle property), 40	attribute), 33 login() (bimmer_connected.api.authentication.MyBMWAuthentication
is_remote_set_ac_limit_enabled (bim-	
mer_connected.vehicle.vehicle.MyBMWVehicle property), 40	
is_remote_set_target_soc_enabled (bim- mer_connected.vehicle.vehicle.MyBMWVehicle	lon (bimmer_connected.models.PointOfInterest at-
property), 40	longitude (bimmer_connected.models.GPSPosition at-
is_remote_unlock_enabled (bim-	
· ·	LOW (bimmer_connected.vehicle.reports.CheckControlStatus attribute), 49
	lsc_type(bimmer_connected.vehicle.vehicle.MyBMWVehicle
mer_connected.vehicle.reports.ConditionBased	
attribute), 50	LscType (class in bimmer_connected.vehicle.vehicle), 38
<pre>is_vehicle_active</pre>	M
property), 40	
is_vehicle_tracking_enabled (bim-	MEDIUM (bimmer_connected.vehicle.reports.CheckControlStatus
mer_connected.vehicle.vehicle.MyBMWVehicle property), 40	messages (bimmer_connected.vehicle.reports.CheckControlMessageReport
isUnlockCableActive (bim-	attribute), 49
mer_connected.models.ChargingSettings	messages (emmes <u>e</u> connected removes open as economical assesses received
attribute), 35	attribute), 50
annome, so	mileage (bimmer_connected.vehicle.vehicle.MyBMWVehicle
L	property), 40 MINI (bimmer_connected.const.CarBrands attribute), 34
lat (bimmer_connected.models.PointOfInterest at-	
tribute), 36	bimmer_connected.account, 30
latitude (bimmer_connected.models.GPSPosition at-	
tribute), 35	bimmer_connected.api.client, 33
Lid (class in bimmer_connected.vehicle.doors_windows),	
45	bimmer_connected.api.utils, 34
lids (bimmer_connected.vehicle.doors_windows.DoorsA	
attribute), 45	bimmer_connected.models, 35
LidState (class in bim-	bimmer_connected.utils, 38
mer_connected.vehicle.doors_windows), 45	bimmer_connected.vehicle.charging_profile,
	es.Servibammer_connected.vehicle.doors_windows,
	cation bimmer_connected.vehicle.fuel_and_battery,
,,, -	bimmer_connected.vehicle.location,48

<pre>bimmer_connected.vehicle.remote_services, 41</pre>	OPEN (bimmer_connected.vehicle.doors_windows.LidState attribute), 46
bimmer_connected.vehicle.reports,48	open_lids (bimmer_connected.vehicle.doors_windows.DoorsAndWindows
bimmer_connected.vehicle.vehicle, 38	property), 45
MyBMWAccount (class in bimmer_connected.account), 30	OPEN_TILT (bimmer_connected.vehicle.doors_windows.LidState
MyBMWAPIError, 35	attribute), 46
MyBMWAuthentication (class in bim-	open_windows (bimmer_connected.vehicle.doors_windows.DoorsAndW
mer_connected.api.authentication), 32	property), 45
MyBMWAuthError, 35	${\tt OVERDUE} (bimmer_connected. vehicle. reports. Condition Based Service Status$
MyBMWClient (class in bimmer_connected.api.client), 33	attribute), 50
<pre>MyBMWClientConfiguration (class in bim- mer_connected.api.client), 33</pre>	P
MyBMWJSONEncoder (class in bimmer_connected.utils), 38	parse_datetime() (in module bim- mer_connected.utils), 38
MyBMWLoginClient (class in bim-	PARTIALLY_LOCKED (bim-
mer_connected.api.authentication), 32	mer_connected.vehicle.doors_windows.LockState
MyBMWLoginRetry (class in bim-	attribute), 46
mer_connected.api.authentication), 32	password (bimmer_connected.account.MyBMWAccount
MyBMWQuotaError, 35	attribute), 31
MyBMWRemoteServiceError, 35	PENDING (bimmer_connected.vehicle.remote_services.ExecutionState
MyBMWVehicle (class in bim-	attribute), 41
mer_connected.vehicle.vehicle), 39	${\tt PENDING} \ (bimmer_connected. vehicle. reports. Condition Based Service Status) \\$
N	attribute), 50
	phoneNumber (bimmer_connected.models.PointOfInterest
name (bimmer_connected.models.PointOfInterest at-	attribute), 36
tribute), 36	PLUGGED_IN (bimmer_connected.vehicle.fuel_and_battery.ChargingState
name (bimmer_connected.vehicle.doors_windows.Lid at-	attribute), 47
tribute), 45	PointOfInterest (class in bimmer_connected.models),
name (bimmer_connected.vehicle.vehicle.MyBMWVehicle	35
property), 40 NO_PRESELECTION (bim-	PointOfInterestAddress (class in bim-
`	mer_connected.models), 36 gbbsfeadn.code (bimmer_connected.models.PointOfInterest
attribute), 43	attribute), 36
	postalCode (bimmer_connected.models.PointOfInterestAddress
tribute), 35	attribute), 37
NOT_CAPABLE (bimmer_connected.vehicle.vehicle.LscType	
attribute), 39	mer_connected.vehicle.charging_profile.ChargingProfile
NOT_CHARGING(bimmer_connected.vehicle.fuel_and_batte	
attribute), 47	provider (bimmer_connected.models.PointOfInterest
NOT_SUPPORTED(bimmer_connected.vehicle.vehicle.LscTy	
attribute), 39	providerId (bimmer_connected.models.PointOfInterest
	attribute), 36
0	${\tt providerPoiId} ({\it bimmer_connected.models.PointOfInterest}$
observer_position (bim-	attribute), 36
mer_connected.account.MyBMWAccount	R
attribute), 31	
observer_position (bim-	refresh_token (bimmer_connected.account.MyBMWAccount
mer_connected.api.client.MyBMWClientConfigur	
attribute), 33 OK (bimmer_connected.vehicle.reports.CheckControlStatus	region (bimmer_connected.account.MyBMWAccount at- tribute), 31
ok (bimmer_connected.venicie.reports.CheckControlsialus attribute), 49	region (bimmer_connected.models.PointOfInterestAddress
anribute), 49 OK (bimmer_connected.vehicle.reports.ConditionBasedServ	
attribute), 50	regionCode (bimmer_connected.models.PointOfInterestAddress
	attribute). 37

```
Regions (class in bimmer connected.const), 34
                                                       set_remote_service_position()
                                                                                                       (bim-
remaining_battery_percent
                                                (bim-
                                                                mer connected.vehicle.location.VehicleLocation
         mer connected.vehicle.fuel and battery.FuelAndBattery
                                                                method), 48
                                                       \verb|settlement| (bimmer\_connected.models.PointOfInterestAddress|
         attribute), 47
remaining_fuel
                                                (bim-
                                                                attribute), 37
         mer connected.vehicle.fuel and battery.FuelAndBaDEr(bimmer connected.vehicle.vehicle.VehicleViewDirection
         attribute), 47
                                                                attribute), 41
remaining_fuel_percent
                                                (bim- software_version
                                                                                                       (bim-
         mer connected.vehicle.fuel and battery.FuelAndBattery
                                                                mer connected.vehicle.reports.Headunit
         attribute), 47
                                                                attribute), 50
remaining_range_electric
                                                (bim-
                                                       sourceType (bimmer_connected.models.PointOfInterest
         mer_connected.vehicle.fuel_and_battery.FuelAndBattery
                                                                attribute), 36
                                                       start_time(bimmer_connected.vehicle.charging_profile.ChargingWindow
         attribute), 48
                                                                property), 44
remaining_range_fuel
                                                (bim-
         mer_connected.vehicle.fuel_and_battery.FuelAndBattery_time (bimmer_connected.vehicle.charging_profile.DepartureTimer
         attribute), 48
                                                                property), 44
                                                (bim- state(bimmer_connected.vehicle.reports.CheckControlMessage
remaining_range_total
         mer_connected.vehicle.fuel_and_battery.FuelAndBattery attribute), 49
         attribute), 48
                                                       \verb+state+ (bimmer\_connected.vehicle.reports.ConditionBasedService
remote_service_position
                                                (bim-
                                                                attribute), 49
         mer\_connected.vehicle.location.VehicleLocation street (bimmer\_connected.models.PointOfInterest at-
         attribute), 48
                                                                tribute), 36
RemoteServices
                                                       street(bimmer\_connected.models.PointOfInterestAddress)
                         (class
                                      in
                                                bim-
         mer connected.vehicle.remote services),
                                                                attribute), 37
                                                       StrEnum (class in bimmer_connected.models), 37
RemoteServiceStatus
                             (class
                                        in
                                                bim-
                                                       sync_auth_flow()
         mer_connected.vehicle.remote_services),
                                                                mer_connected.api.authentication.MyBMWAuthentication
                                                                method), 32
REST_OF_WORLD (bimmer_connected.const.Regions at-
                                                       sync_auth_flow()
                                                                                                       (bim-
                                                                mer_connected.api.authentication.MyBMWLoginRetry
         tribute), 35
                                                                method), 32
S
SECURED (bimmer_connected.vehicle.doors_windows.LockState
         attribute), 46
                                                       TARGET REACHED
                                                                                                       (bim-
SELECTIVE_LOCKED
                                                (bim-
                                                                mer_connected.vehicle.fuel_and_battery.ChargingState
         mer connected.vehicle.doors windows.LockState
                                                                attribute), 47
         attribute), 46
                                                       timer_id(bimmer_connected.vehicle.charging_profile.DepartureTimer
{\tt SEND\_POI}\ (bimmer\_connected.vehicle.remote\_services. Services
                                                                property), 44
         attribute), 43
                                                       timer_type(bimmer connected.vehicle.charging profile.ChargingProfile
service_type(bimmer_connected.vehicle.reports.ConditionBasedSaxwiribute), 44
         attribute), 49
                                                       TimerTypes
                                                                             (class
                                                                                                        bim-
                    (class
                                                                mer_connected.vehicle.charging_profile),
Services
                                                bim-
                                    in
         mer_connected.vehicle.remote_services),
         42
                                                       timestamp(bimmer_connected.vehicle.vehicle.MyBMWVehicle
set_log_responses()
                                                (bim-
                                                                property), 40
         mer_connected.api.client.MyBMWClientConfigurationcamel_case() (in module bimmer_connected.utils),
         method), 33
                                                                38
set_observer_position()
                                                (bim-
                                                       trigger_charge_start()
                                                                                                       (bim-
                                                                mer_connected.vehicle.remote_services.RemoteServices
         mer_connected.account.MyBMWAccount
         method), 31
                                                                method), 41
set_refresh_token()
                                                (bim-
                                                       trigger_charge_stop()
                                                                                                       (bim-
         mer_connected.account.MyBMWAccount
                                                                mer_connected.vehicle.remote_services.RemoteServices
         method), 31
                                                                method), 41
```

```
trigger_charging_profile_update()
                                                                           (bim- UNKNOWN (bimmer_connected.vehicle.fuel_and_battery.ChargingState
              mer\_connected.vehicle.remote\_services.RemoteServices
                                                                                                    attribute), 47
              method), 42
                                                                                      UNKNOWN (bimmer connected.vehicle.remote services.ExecutionState
                                                                           (bim-
trigger_charging_settings_update()
                                                                                                     attribute), 41
              mer_connected.vehicle.remote_services.RemoteSe\(\formall \) (bimmer_connected.vehicle.reports.CheckControlStatus
              method), 42
                                                                                                    attribute), 49
trigger_remote_air_conditioning()
                                                                           (bim- UNKNOWN (bimmer connected.vehicle.reports.ConditionBasedServiceStatus
              mer connected.vehicle.remote services.RemoteServices
                                                                                                    attribute), 50
              method), 42
                                                                                      UNKNOWN (bimmer connected.vehicle.vehicle.LscType at-
trigger_remote_air_conditioning_stop() (bim-
                                                                                                     tribute), 39
              mer_connected.vehicle.remote_services.RemoteSe\textbf{VNKNOWN} (bimmer_connected.vehicle.vehicle.VehicleViewDirection
              method), 42
                                                                                                     attribute), 41
trigger_remote_door_lock()
                                                                           (bim- UNLOCKED (bimmer_connected.vehicle.doors_windows.LockState
              mer_connected.vehicle.remote_services.RemoteServices
                                                                                                    attribute), 46
                                                                                      update_from_vehicle_data()
                                                                                                                                                                 (bim-
              method), 42
trigger_remote_door_unlock()
                                                                           (bim-
                                                                                                     mer_connected.models.VehicleDataBase
              mer_connected.vehicle.remote_services.RemoteServices
                                                                                                    method), 37
              method), 42
                                                                                      update_state()
                                                                                                                                                                 (bim-
                                                                           (bim-
                                                                                                    mer_connected.vehicle.vehicle.MyBMWVehicle
trigger_remote_horn()
              mer connected.vehicle.remote services.RemoteServices
                                                                                                    method), 41
              method), 42
                                                                                      use_metric_units
                                                                                                                                                                 (bim-
trigger_remote_light_flash()
                                                                           (bim-
                                                                                                    mer connected.account.MyBMWAccount
              mer\_connected.vehicle.remote\_services.RemoteServices
                                                                                                    attribute), 31
              method), 42
                                                                                      username (bimmer connected.account.MyBMWAccount
trigger_remote_service()
                                                                           (bim-
                                                                                                    attribute), 31
              mer_connected.vehicle.remote_services.RemoteServices
              method), 42
trigger_remote_vehicle_finder()
                                                                           (bim- valid_regions()
                                                                                                                                            module
                                                                                                                                                                  bim-
              mer_connected.vehicle.remote_services.RemoteServices
                                                                                                    mer_connected.api.regions), 33
              method), 42
                                                                                      value
                                                                                                   (bimmer_connected.models.ValueWithUnit
                                                                           (bim-
trigger_send_poi()
                                                                                                    tribute), 37
              mer\_connected. vehicle. remote\_services. Remote Several With Unit (class in bimmer\_connected. models), 37 in the connected of the connected 
              method), 42
                                                                                      VEHICLE_FINDER
                                                                                                                                                                 (him-
try_import_pillow_image()
                                                            module
                                                                           bim-
                                                   (in
                                                                                                    mer connected.vehicle.remote services.Services
              mer connected.api.utils), 34
                                                                                                    attribute), 43
TWO_TIMES_TIMER
                                                                           (bim-
                                                                                      vehicle_update_timestamp
              mer connected.vehicle.charging profile.TimerTypes
                                                                                                    mer_connected.vehicle.location.VehicleLocation
              attribute), 44
                                                                                                    attribute), 48
           (bimmer_connected.models.PointOfInterest
type
                                                                                      vehicleCategoryId
                                                                                                                                                                 (bim-
              tribute), 36
                                                                                                    mer_connected.models.PointOfInterest
                                                                                                                                                                     at-
                                                                                                    tribute), 36
U
                                                                                      VehicleDataBase (class in bimmer_connected.models),
            (bimmer_connected.models.ValueWithUnit
                                                                                                    37
                                                                                      VehicleLocation
                                                                                                                              (class
                                                                                                                                                                  bim-
              tribute), 37
                                                                                                                                                   in
UNKNOWN (bimmer_connected.vehicle.charging_profile.ChargingModemer_connected.vehicle.location), 48
                                                                                      vehicles (bimmer_connected.account.MyBMWAccount
              attribute), 43
UNKNOWN (bimmer_connected.vehicle.charging_profile.ChargingPreference), 31
                                                                                      VehicleViewDirection
                                                                                                                                                      in
                                                                                                                                                                  bim-
              attribute), 43
                                                                                                                                     (class
                                                                                                    mer connected.vehicle.vehicle), 41
UNKNOWN (bimmer_connected.vehicle.charging_profile.TimerTypes
              attribute), 45
                                                                                      vin (bimmer connected.vehicle.vehicle.MyBMWVehicle
                                                                                                    property), 41
UNKNOWN (bimmer_connected.vehicle.doors_windows.LidState
              attribute), 46
UNKNOWN (bimmer_connected.vehicle.doors_windows.LockSM)
              attribute), 46
                                                                                      WAITING_FOR_CHARGING
                                                                                                                                                                 (bim-
```

```
mer_connected.vehicle.fuel_and_battery.ChargingState attribute), 47

weekdays (bimmer_connected.vehicle.charging_profile.DepartureTimer property), 44

WEEKLY_PLANNER (bimmer_connected.vehicle.charging_profile.TimerTypes attribute), 45

Window (class in bimmer_connected.vehicle.doors_windows), 46

windows (bimmer_connected.vehicle.doors_windows.DoorsAndWindows attribute), 45
```