UNICAMP

Project COPEL:ANEEL - PD-02866-0508/2019
Open Middleware and Energy Management System for the House of the Future

Normative Documentation
Middleware System Applied in HEMS

Version Final

Preparation: 1. Ruan dos S. Carvalho.
2. Luiz Carlos B. C. Ferreira.

University State of Campinas - FEEC

December 6, 2022

Contents

List of Figures

List of Tables

1 National Preface

The Brazilian Association of Technical Standards (ABNT) is the National Forum for Standardization. The Brazilian Standards, whose content is the responsibility of the Brazilian Committees (ABNT/CB), the Sectorial Standardization Bodies (ABNT/ONS) and the Special Temporary Study Commissions (ABNT/CEET), are prepared by Study Commissions(CE), formed by representatives of the sectors involved, including: producers, consumers and neutrals (universities, laboratories and others).

Brazilian Association of Technical Standards (ABNT) draws attention to the possibility that some of the elements of this document may be the subject of patent rights. Identification of any patent rights by third parties, this must be communicated to ABNT at any time (Law nº 9.279, of May 14, 1996).

ABNT Technical Documents, as well as International Standards (ISO and IEC), are voluntary and do not include contractual, legal or statutory requirements. ABNT Technical Documents do not replace Laws, Decrees or Regulations, which users must comply with, taking precedence over any ABNT Technical Document.

It should be noted that the ABNT Technical Documents may be cited in Technical Regulations. In these cases, the bodies responsible for the Technical Regulations can determine the dates for demanding the requirements of any ABNT Technical Documents.

The is standard was prepared by the research project of the communication department of the faculty of electrical engineering and computing of the state university of campinas (Unicamp/FEEC), in partnership with the energy company of Paraná (Copel) and the hardware design department of the Eldorado research institute, under the number: Aneel - PD-02866-0508/201.

The standard, under the general heading “System middleware normative document applied in HEMS”. This document contains: Data encoding, Rest API application in Python language, Data transmission protocols and specifications and in HEMS, middleware architecture, Web Services, APIs REST Django Rest framework environment.

2 Scope

The is standard establishes a system architecture, which allows an implementation in a specified middleware, through REST APIs microservices, for an energy management system, in addition to the standards that apply to the microservices profiles, for internet of things (IoT), HEMS, which include the Web Services server specified in this document, and Rest API systems and their supported functionality.

NOTE The specification of these services for the HEMS middleware guarantees the interoperability of applications in different implementations of platforms that support it.

This Standard provides the specification of the Web Services provided by the Django Rest framework environment and the definitions of the APIs that implement these services.

This Standard does not specify how the ”System middleware normative document applied in HEMS” is implemented in a compliant receiver, only the requirements for such compliance to be achieved by all implementations.

2.1 Normative References

The following documents are cited in the text in such a way that their contents, in whole or in part, constitute requirements for this document. For dated references, only the cited editions apply. For undated references, the most recent editions of the referenced document (including amendments) apply.

ISO/IEC 30141, *Internet of things (IoT) - Reference architecture*

ISO/IEC 27000, *Information technology — Security techniques — Information security management systems — Overview and vocabulary;*

ISO/IEC 17963: 2013, *Web Services for Management (WS-Management) Specification*

ISO/IEC 27033-6: 2016, *Information technology — Security techniques — Network security — Securing wireless IP network access*

IEC 61588: 2009, *Precision clock synchronization protocol for networked measurement and control system*

IETF RFC 7252:2014, *The Constrained Application Protocol (CoAP)*

ISO/IEC/IEEE 8802-15-4:2010,*Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 15-4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)*

ISO/IEC 20922:2016, *Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1*

ISO/IEC 30118-17:2021, *Information technology – Open Connectivity Foundation (OCF) Specification — Part 17: OCF resource to Zigbee cluster mapping specification*

ISO/IEC 20547-3,*Information technology — Big data reference architecture — Part 3: Reference architecture.*

ISO/IEC 29341-30-2: 2017,*Information technology — UPnP Device Architecture: IoT management and control device control protocol — IoT management and control device*

ISO/IEC TR 13066-2: 2016, *Information technology — Interoperability with assistive technology (AT) — Windows accessibility application programming interface (API)*

2.2 Terms and Definitions

For the purposes of this normative document, the following terms and definitions apply.

2.3 Broadband Environment

Environment composed of resources, content, applications and Internet services, reachable through a broadband connection, always active.

2.4 Local Environment

Environment composed of resources and devices connected to the home network and their respective application execution platforms, which commonly include receivers, smartphone platforms and other systems on the electrical network

2.5 System Application

Information that expresses a specific set of observable behaviors.

2.6 Wi-SUN (Wireless Smart Utility Network)

It is a wireless communication standard that enables seamless connectivity between smart-grid devices. The standard powers large-scale outdoor IoT networks like wireless mesh networks for Advanced Metering Infrastructure (AMI), home energy management, distribution automation, and other large scale outdoor network applications including FAN (Field Area Networks) and HAN (Home Area Networks).

2.7 Application Programming Interface API

Set of well-defined methods, functions, protocols, routines or commands which application software uses with facilities of programming languages to invoke services.

2.8 Web Services

Technology used in systems integration and communication between different applications, with the objective of making resources of software environments available over the network in a standardized way.

2.9 MQTT Protocol

Connectivity protocol designed as an extremely lightweight publish/subscribe messaging transport.

NOTE to entry: It is standardised by the Advancing Open Standards for the Information Society (OASIS).

2.10 Representational State Transfer REST

REST describes a machine to interface In web development, REST allows content to be rendered when requested, often referred to as Dynamic Content.

2.11 JavaScript Object Notation

JSON is a text-based, schema-free representation based on key-value pairs and ordered lists. Although JSON is derived from JavaScript, it is supported natively or through libraries in most major programming languages.

2.12 Deep link

Deep linking is the use of a hyperlink that links to specific, usually searchable or indexed, web content on a website, rather than the home page of the website. The URL contains all the information needed to point to a particular item.

2.13 Token

Strings, numbers, literals, or one of six structural characters

NOTE to entry: The six structural characters are “”, “”, “ [”, “] ”, “:”, “, ”.

2.14 JMeter

Apache JMeterTM is pure Java open source software designed to test load test functional behavior and measure performance. This software may be used to analyze and measure the performance of the web application or a variety of services.

2.15 Django Rest Framework

Django REST Framework or DRF is a library that allows building REST APIs using the Django framework, because it works under the Django framework, allows the construction of APIs on any platform.

NOTE to entry: Generic classes for operations CRUD (Create, Read, Update, Delete).

2.16 SQlite

SQLite is a relational database that, unlike other tools of its type, does not store information on a server. This independence happens because he can put his files inside himself.

2.17 Python Language

Python is a high-level programming language or High Level Language dynamic, interpreted, modular, multi platform and object oriented a specific way of organizing software where procedures are submitted to classes, which allows greater control and stability of code for large projects.

2.18 Hardware

NXP i.Mx6 microprocessor, with 396 MHz clock, 512MB of RAM and 32 GB of storage, running a real time Linux operating system, based on the Debian distribution, for embedded systems. Smart outlets use the WI-SUN HAN protocol to communicate with an NXP card, where the middleware is running.

2.19 microservice

Independently deployable artefact providing a service implementing a specific functional part of an application.

2.20 WebSocket

Protocol which enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

2.21 RESTful API description language

Language designed to provide a structured description of a RESTful Web API that is useful both to humans and for automated machine processing.

2.22 Zigbee

unique device identifier and a set of mandatory and optional clusters to be implemented on a single Zigbee endpoint

2.23 List of Acronyms

For the purposes of this standard, the following abbreviations apply.

Acronyms	**Meaning**
API	Application Programming Interface.
CoAP	Constrained Application Protocol.
CRUD	Create, Read, Update, Delete.
DRF	Django REST framework.
HEMS	Home Energy Management System.
HTTP	Hypertext Transfer Protocol.
IOT	Internet of Things.
JSON	JavaScript Object Notation.
MQTT	Message Queue Telemetry Transport.
ORM	Object Relational Mapper.
REST	Representational State Transfer.
SQL	Structured Query Language.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
URL	Uniform Resource Lacator.
WI-FI	Wireless Fidelity.
Wi-SUM	Wireless Smart Utility Network.
HAN	Home Area Networks.

3 Basic System Architecture

3.1 Middleware System

Architecture has core that is formed by native modules. These modules have the basic functions for their operation. The modules are independent processes, each performing its defined and specific tasks. All the functions offered by the modules are accessible through the microservices, using REST APIs. Each micro-service has specific and objective functions that provide full control and configuration of the *middleware*. The structure of the application environment must be in accordance with figure 3.1

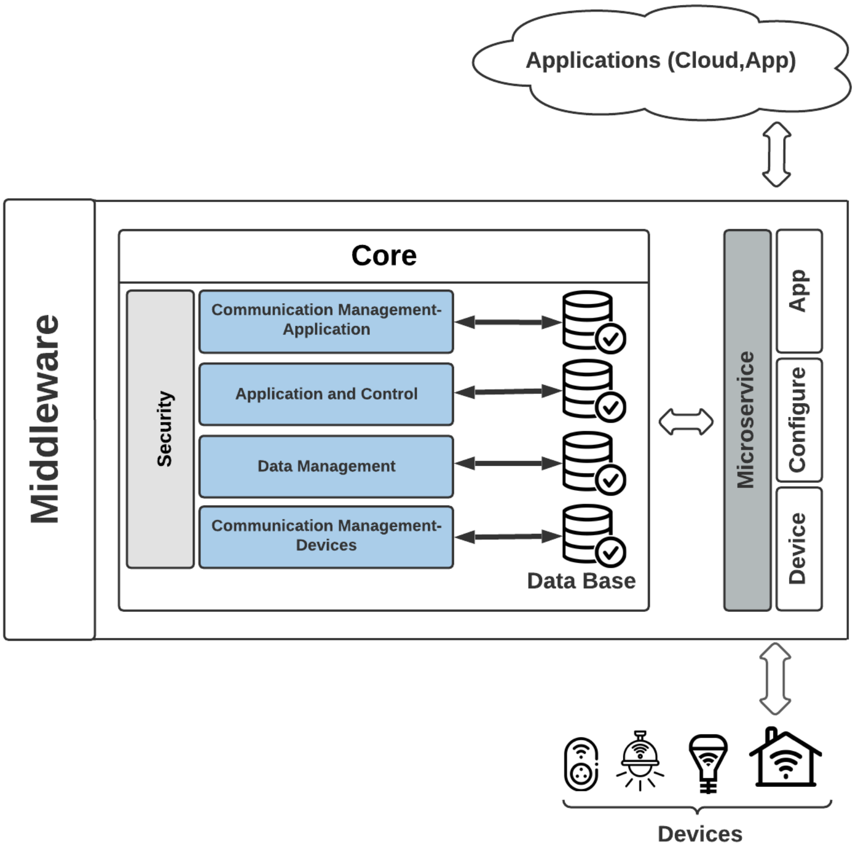

Figure 3.1: **Middleware Architecture**

3.2 Module Specification

The middleware architecture must conform to the following structural elements:

3.3 Microservices:

The microservices are composed of local communication APIs that use well-defined network protocols, in order to allow the request of data requests by the other blocks of the middleware. This is done through a set of commands that abstract these protocols and remote communication APIs that use existing protocols for this purpose, such as MQTT, TCP/UDP, HTTPS, REST, in order to provide a suite of send and receive commands. of information, the microservices independence improves middlewares resiliency and performance, reducing concurrent and parallel accesses with independent databases.

- **App:** Responsible for communication with the application
- **Cloud:** Responsible for communication with the cloud
- **Devices:** Responsible for communication with devices (outlets)
- **Configure:** Responsible for the main settings of the Middleware

3.4 Communication Management - Application:

The module works with receiving and sending data, which meet the requests for writing and reading data in the cloud or local storage. These requests are received through remoting APIs and data management mechanisms. There is the identification of cloud resources, which verify that the online services are working properly. Several tools and protocols are available for sending local data to remote repositories. Many of these options are available in this module. It is possible to send local data via MQTT protocol, which is widely employed in the IoT scenario. Additionally, it is possible to employ HTTP using the CoAP.

3.5 Control and Application Management :

Middleware provides an environment for user interaction with active devices on the local network, organized into simplified, user-specified programming routines. This module includes services to control devices such as ON/OFF control, dimmer and green power controllers. The system allows the development of various applications for HEMS and smart home environment.

3.6 Data Management:

This module processes input data from HEMS devices, which work with storage and filtering. The focus of this module is local data storage. Several applications can generate data in HEMS, requiring a reliable storage system. Database size must be managed properly to avoid middleware performance degradation. The local database is designed as a circular queue, where data is constantly updated with the latest data. Circular queue volume is configurable. The local database stores information for a short period of time, as all data is sent to remote repositories to be consolidated and processed. Below are some characteristics.

a) **Integrity:** There is a verification and correction, if necessary, of the data received by the devices through the APIs.

b) **Local storage:** The data can be useful in situations where the cloud server is unavailable at the time of data collection, allowing the data to be sent after the connection is reestablished.

c) **Cloud Storage:** Processed data will be transferred to the cloud through APIs.

d) **Extracting and adding data to the database:** Requesting data to the middleware using APIs and managing remoting. Addition of the data after passing through the integrity check and eventual processing.

3.7 Communication Management - Devices:

This component manages the communication between the HEMS controller and local devices. It has multiple communication interfaces and is accessed and configured through microservices APIs. The choice of communication protocol is based on issues related to the communication network, such as distance between nodes, transmission capacity and quality of the necessary link. The middleware must be flexible enough to support different protocols such as Wi-SUN, Wi-Fi, which are commonly adopted standards.

3.8 Security:

This component deals with security in the various middleware components, from data reception to sending it to remote applications. It is important to mention that there is not only one security tool capable of covering all components, therefore, it is necessary to define techniques individual safety data for the various components:

a) The communication layer among HEMS devices, used to send data to the client applications, security is provided by the communication protocols employed, such as Wi-SUN, Wi-Fi,and Zigbee, which already have their own security layers.

b) The operating system level, security is based on access control, encryption, and hash algorithms for reliable and confidential data storage.

c) Application protocols used to send local data to remote repositories, such as HTTP, MQTT, and CoAP, also have their own security mechanisms.

d) Finally, the microservices architecture encapsulates the functions, making the source code inaccessible to users.

4 Middleware Implementation

The modules are implemented independently. This approach generates flexibility and better organization, as functions can be implemented in a specific module, without worrying about the other modules.

The Python language offers this feature, that is, to create custom modules in an application.

Figure 4.1 shows the organization of middleware modules. In addition to the organization of the modules, there is an organization for the use of the functions of each module. Each running module is treated as an independent process in the operating system.

To achieve the independence of the modules in the core, the concept of multiprocessing was employed, wherein each module is treated as an independent process. Therefore, the main middleware file, named orchestrator *mdw-orq.py*, is responsible for creating and managing these processes. The orchestrator initiates and manages the parallel execution of the middleware modules

This modular organization and processes running in parallel provide middleware flexibility.

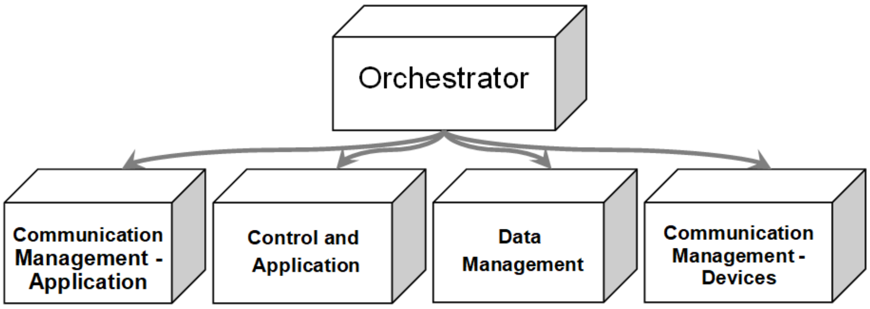
Orchestrator
Communication Management - Application
Control and Application
Data Management
Communication Management - Devices

Figure 4.1: **Middleware Orchestrator**

Microservices perform a key role in the proposed middleware. They provide functions enabling the consumption of the generated data, in addition to saving and modifying information in the databases in a independent and scalable way.

The APIs provide the capability of changing the behavior of the middleware in a standardized way via the REST pattern.

4.1 Communication Management - Application:

This module implements functions to send data to the cloud computing software, through an Internet connection. Presently, two options are available for transferring data: (I) sending to an IoT Hub (a paid online repository supported by Microsoft); (II) sending to any MQTT Broker. Hence, the following functions are implemented:

- **Search_data:** This function is responsible for database queries for preparing the information to be sent to the cloud computing software.
- **Send_IotHub:** Responsible for sending the data to the IoT Hub.
- **Send_MQTT:** Responsible for sending the data to the MQTT Broker.
- **Resend_data:** This function is responsible for searching the data whose transmissions failed, and attempting to resend them.

The destination of data is predefined by default, although it is possible to modify it via the APIs. The data is periodically uploaded to the cloud computing software, with the upload period adjustable via the corresponding microservice.
Figure 4.2, shows the communication management flowchart with the application.

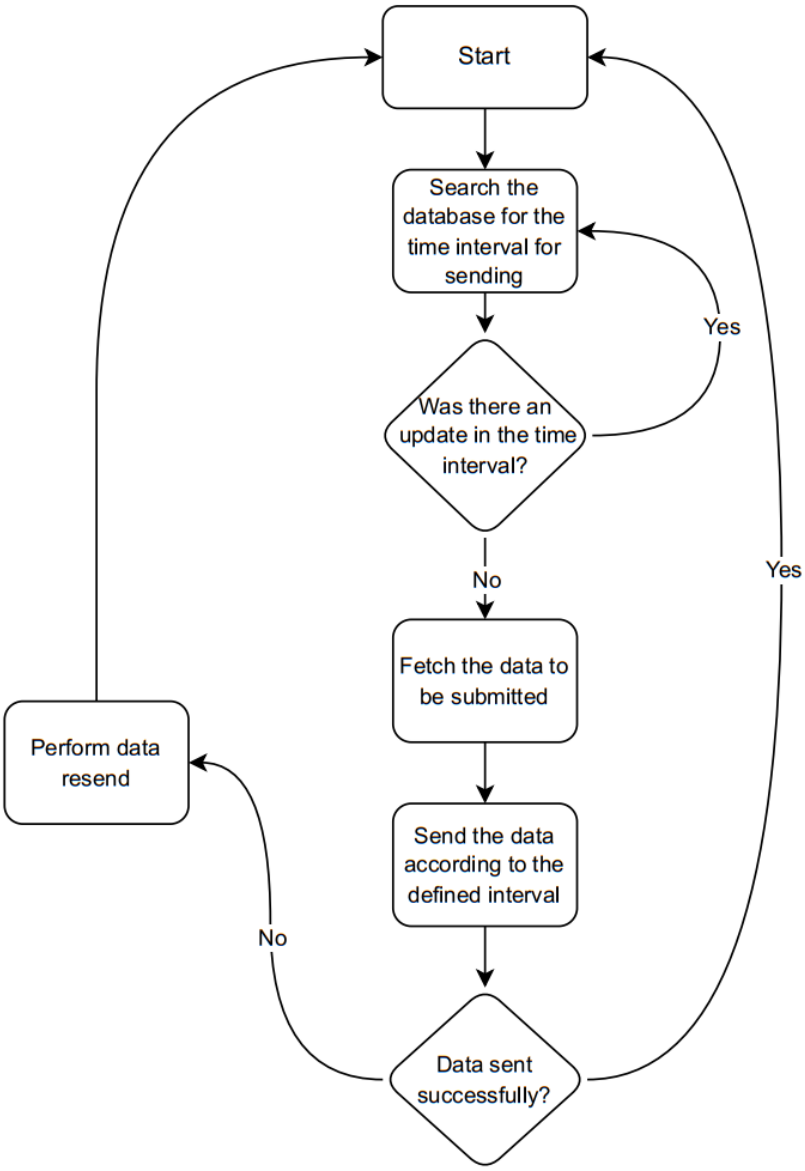
Start
Search the database for the time interval for sending
Yes
Was there an update in the time interval?
No
Yes
Fetch the data to be submitted
Perform data resend
Send the data according to the defined interval
No
Data sent successfully?

Figure 4.2: **Communication Flowchart with the Application**

4.2 Data management:

This module implements the management of the middleware databases. By default, there are four databases for these types of information:

- Raw data collected from the devices.
- Information genereted by processing the raw data.
- Configuration data for middleware operation.
- HEMS identification data.

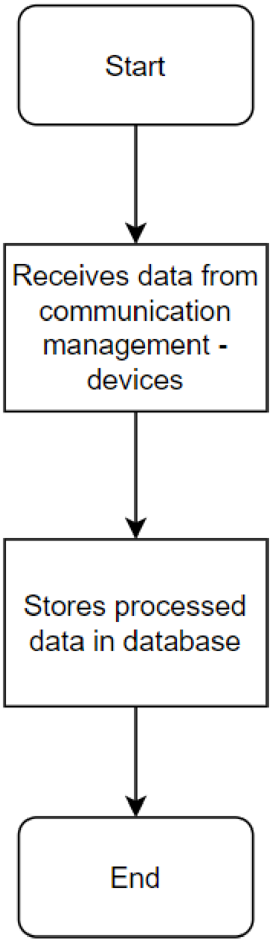
Start
Receives data from communication management - devices
Stores processed data in database
End

Figure 4.3: **Flowchart data management**

These databases are accessed by all middleware modules, being connecting points among the modules. Therefore, the modules are responsible for the exchange of information among the module and are accessible via microservices.

The database responsible for storing the raw data collected from the devices is implemented using a circular buffer format to achieve an ordered expansion and avoid exceeding a prespecified database size.

The microservices provide functions to handle the data, enabling data collection and changing middleware operation parameters, affording flexibility and standardized access.

4.3 Devices communication management:

The Device Communication Management Module contains functions for interface configuration and scheduling of data reading and storage.

Figure 4.4 shows the flowchart of the current module. Where device management receives the data for the outlets and sends it so that the data management can carry out the processing of the same.

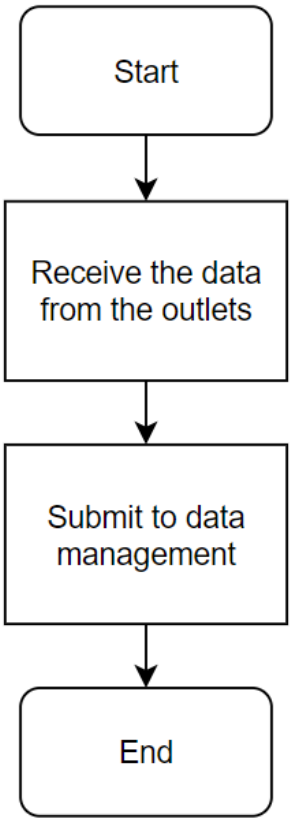

Figure 4.4: **Device Management Flowchart.**

The middleware communicates with devices (e.g., smart outlets and smart meters) according to a template defined for each appliance. Different templates are available for implementation, allowing communication via several protocols. In our work, the data are collected from the files generated by the smart outlets. The microservice allows the registration of the template, containing the required information to access the device data. The template is presented in JSON format, as depicted in listing 4.1.

listing 4.1.

```
{
        "device":
                {
                        "Type": "outlet",
                        "Patch": "/users/measures",
                        "Id/MAC": "000e2ef48448",
                        "Data": "[voltage,current, active_power,
                        reactive_power, power_factor, aparent_power]",
                        "Protocol": "Wisun-han",

                }

}
```

When a device adopts the CoAP protocol or supports the HTTP standard, an alternative way to collect data is based on the microservice named receive_data. In this case, the device sends the data via a POST or PUT request. This alternative solution provides interoperability between the middleware and the devices.

The functions of the module are:

- **Device_Template():** This function stores and captures the format of the information provided by the devices, besides the information from the device itself.
- **Time_Request():** It sets the time between data requests to the device.
- **Config_Interface():** It stores and captures information regarding the communication interfaces.
- **Receive_data():** It checks for incoming data from devices.
- **Store_data():** This function stores the received information in the database.

All functions are accessible via microservices. Therefore, using an mobile App or Web interface, an administrator is allowed to alter the middleware configuration.

4.4 Security Management:

The middleware security is implemented through the security protocols provided by the solutions and protocols employed in communication, data storage, and cloud computing.

- **Access Control:** Only registered users are allowed to access the APIs,
- **Data Sending Authentication:** All data are sent to the cloud computing software using authentication at the MQTT broker,
- **Microservices:** The microservice architecture inherently encapsulates the functions, not allowing direct access to any portion of the system.

4.5 Microservices (APIs REST)

Microservices offer the functions provided by the middleware based on the REST standard. Complete documentation on the APIs is available to users and developers. The use of Rest standard allows any entity to interact with the middleware.

Each microservice performs specific functions and maintains a different database. The microservices are organized as follows:

- **App:** Responsible for the communication with cloud computing software and the client applications,
- **Cloud:** Responsible for communication with the cloud
- **Devices:** Responsible for the communication with the devices (outlets),
- **Configure:** Responsible for the middleware configuration.

A brief description of each microservice is presented in the following paragraphs.

4.6 API communication with application (App)

This microservice is responsible for the communication with the client applications. It contains four entities: Device, Hems_sys, Zone, and outlet. The Device entity is the table responsible for registering devices and relies on three fields in Django: the device name, whether the device is active or not, and whether it is a generating or consuming device. The Hems_sys entity contains information of the user that must be stored, both in the cloud and in the middleware, providing easier access to it. Therefore, this entity includes the following fields:

The device entity (Table 5.1) is the table responsible for registering the devices. For this reason it has three fields in Django:

- **devName:** device name (eg television, refrigerator, etc.)
- **devType:** identifies the device type (generation or consumption)
- **dev_on:** informs whether the socket to which the device is connected is turned on or not.

Table 4.1: **Device**

variable	Data types in Django/SQlite
devName	models.CharField()
devType	models.CharField()
dev_on	models.BooleanField()

The following table, Hems_sys (Table 5.2), is where you have the user information that must be saved both in the cloud and in the middleware itself, in order to facilitate their access.

- **hems_reg_date:** date and time the HEMS was registered,
- **userName:** User name,
- **priceKWh:** price in kWh ,
- **hems_last_update:** HEMS latest update,
- **homeCity:** City of HEMS.
- **homeStreet:** Address and HEMS installation number;
- **home Neighbour:**Neighborhood in which HEMS is located;
- **homeComplement:** Complement HEMS address.

Table 4.2: **Hems_sys**

variable	Data types in Django/SQlite
hems_reg_date	models.DateTimeField()
userName	models.CharField()
priceKWh	models.DecimalField()
hems_last_update	models.DateTimeField()
homeCity	models.CharField()
homeStreet	models.CharField()
homeNeighbour	models.CharField()
homeComplement	models.CharField()

The entities Zone (table 5.3) and Outlet (table 5.4) refer to the household area where the outlet is installed (living room, kitchen, etc.) and a set of information on the registered outlets, respectively. These set o information includes the outlet type (relay or dimmer), which appliance is connected to, allowing the identification of the outlet.

In the Zone entity we have the following fields:

- **name:** Name of the rooms in the house where devices can be found.

Table 4.3: **Zone**

variable	Data types in Django/SQlite
Name	models.CharField()

The Outlet entity provides more detailed information:

- **label:** Socket Identifier;
- **outletType:** identifies whether the device is a relay or dimmer;
- **connectedDevice_id:** Identifier of the socket that is connected (Link with the Data entity of api_devices);
- **outletZone_id:**Identifier of the room to which that outlet belongs (Link with the Zone entity);

Table 4.4: **Outlet**

variable	Data types in Django/SQlite
label	models.CharField()
outletType	models.CharField()
connectedDevice_id	models.ForeignKey()
outletZone_id	models.ForeignKey()

4.7 API cloud communication

This API was named api_cloud because it brings the features that pertain to communication with the cloud. This API has only one unit, Initial_config, which contains the settings for the time interval that information will be sent to the a cloud. It is worth mentioning that when we deal with APIs, we have the flexibility to use their models in other applications, that is, we can use the entities created for the application in the cloud with just a few adjustments depending on the need.
Table 5.5 shows the functions, which are:

- **cloud_interval:** Receives an integer representing the time (in minutes) for the periodic sending of data to the cloud;

- **device_interval:** Receives an integer representing the time (in seconds) for periodic communication with devices, if they allow this configuration

Table 4.5: **Config Initial**

variable	Data types in Django/SQlite
cloud_interval	models.IntegerField()
device_interval	models.IntegerField()

4.8 API - Devices Microservice

The API for communicating with devices, or api_devices as it was named, has five units. The first unit refers to a test communication using the Zigbee interface (Table 5.6) which can be useful in communicating with devices and for future applications in which the use of middleware is explored.

- **port:** Communication port with ZigBee;
- **pan_id:** Network identifier;

Table 4.6: **Zigbee_Interface**

variable	Data types in Django/SQlite
port	models.CharField()
pan_id	models.CharField()

The first is the *Data* entity (Table 5.7) and then the *rootData* (Table 5.8) which brings the information collected from the socket and the *smart meter* and organized in a JSON format as required by the API it self. Thus, it is possible to have more mobility for changes to the variables and also for their presentation in the application

It is a microservice for communicating with the devices and contains two different entities: Data and Root Data. The Data entity addresses the information proceeding from the smart outlets. It includes the following fields:

- **dev_id:** socket identifier,
- **voltage:** Voltage reading by the outlet,
- **current:** Current reading of the outlet,
- **active_power:** Active power reading of the outlet,
- **reactive_power:** Reactive power of the outlet,
- **power_factor:** Power factor of the outlet,

- **device_energy:** Total power of the outlet,
- **time:** Timestamp of the measures.

Table 4.7: **Data**

Variável	**data type in Django/SQlite**
dev_id	models.ForeignKey
voltage	models.CharField
current	models.CharField
active_power	models.CharField
reactive_power	models.CharField
dev_energy	models.CharField
dev_on	models.BooleanField
time	models.DateTimeField

- **time:** Data and time when the power readings were taken;
- **power:** Measurement of the total power used by the house.

Table 4.8: **Root date**

variable	**Data types in Django/SQlite**
time	models.DateTimeField()
power	models.CharField()

The Data entity is accessible via a GET or POST request, allowing the devices to send the data using the CoAP standard or any other standard that supports HTTP requests. This feature provides interoperability with multiple existing devices.

The Root Data entity contains smart meter information such as time (date and time of the measurements) and power (the total kWh value). It is noteworthy that all APIs provide the collected information from the outlet organized in JSON format, allowing flexibility when manipulating the variables, via either the application or the cloud computing software.

In addition to the APIs for communicating with the application, APIs were created that control the sockets and their interface will be through the Application, that is, these APIs will bridge the gap between the application and devices. For this purpose, two entities were created.

to perform this control, the *PlugRequest* (Table 5.9) and the *PlugResponse* (Table 5.10). The first entity is responsible for receiving the data from the application and sending the commands to the socket while the second entity will receive the data. provided by the sockets and return the values to the application.

a) command used to check whether the output is on or not, since the Outlet may control the switched output by timing or scheduling and the periodic sending of measurements may be disabled;

b) turn the outlet on and off, a command that does not generate a response but will reset the outlet's hardware and reconnect with the controller

c) command that will program the state of the switched output when connecting the socket.

Table 4.9: **Plug Request**

variable	Data types in Django/SQlite
readRelayStatus	models.BooleanField()
onOff	models.BooleanField()
onOffTime	models.IntegerField()
reset	models.BooleanField()
switchedOutput	models.CharField()
plugId	models.CharField()

Table 4.10: **Plug Response**

variable	Data types in Django/SQlite
responseId	models.IntegerField()
relayStatus	models.BooleanField()
onOffStatus	models.BooleanField()
switchedOutputStatus	models.BooleanField()

4.9 Configure Microservice

It is used to configure the middleware and contains the following entities:

- **Data request time:** it is the interval within which measurements of the devices are requested;
- **Send-to-cloud method:** it configures the destination and the method to send the data to the cloud;
- **Update:** used to configure the repository where the periodic monitoring for middleware updates is performed;
- **Time-to-cloud:** it sets the time interval for sending information to the cloud.
- **Username and password for MQTT broker:** it sets the credentials for the use of MQTT broker.
- **MQTT Address:** it sets the address of the MQTT Broker.

Each microservice features specific functions and accesses a unique database. This characteristic of microservices ensures better resilience and performance of the middleware since the reading and writing processes occur independently, thus avoiding a large number of simultaneous accesses to a single database. Furthermore, new services can be added, without interfering other services already implemented.

5 Features of REST APIs

A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. This data can be used for GET, PUT, PATCH, POST, and DELETE data types, which refer to reading, updating, creating, and deleting resource-related operations.

The API describes the proper way for a developer to write a program requesting services from an operating system or other application. A RESTful API - also known as a RESTful web service or REST API - is based on representational state transfer (REST), which is an architectural style and approach to communications often used in development of web services.

A RESTful API breaks down a transaction to create a series of small modules. Each module addresses an underlying part of the transaction. This modularity gives developers a lot of flexibility. Django REST is a framework that uses the patterns of a REST architecture to build websites with applications RESTful.

Due to the fact that some of the collected data is vector in nature, it is interesting to use the JSON format, which simplifies the modeling of non-SQL data in a predominantly relational database.

The fact that some data collected is vector in nature, it is interesting to use the JSON format, which simplifies the modeling of non-SQL data in a predominantly relational database. In short, APIs are a set of functions established by software for the use of its functionalities by applications where one does not intend to get involved in details of the software implementation, but only to use its services.

5.1 Data List

The API specification of services and applications is found in the tables below.

5.1.1 GET - datalist/

Table 5.1: **GET/ datalist/**

Request Format:	http(s)://mdw-api/v1/datalist/
Operation Type:	GET
Operation Id:	datalist_list
Description:	• dev • voltage • current • active_power • reactive_power • power_factor • dev_energy • time • id • date_lte
Parameters:	string number (query)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”**dev**”: 0,
	”**voltage**”: ”string”,
	”**current**”: ”string”,
	”**active_power**”: ”string”,
	”**reactive_power**”: ”string”,
	”**power_factor**”: ”string”,
	”**dev_energy**”: ”string”,
	”**time**”: ”2022-11-17T03:47:12.620Z”
	}
	]

Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	Basic

5.1.2 POST - datalist/

Table 5.2: **POST/ datalist/**

Request Format:	http(s)://mdw-api/v1/datalist/
Operation Type:	POST
Operation Id:	datalist_create
Description:	• voltage • current • active_power • reactive_power • power_factor • dev_energy • time
Parameters:	data object (body) **required:** *true*
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	[
	{
	”**dev**”: 0,
	”**voltage**”: ”string”,
	”**current**”: ”string”,
	”**active_power**”: ”string”,
	”**reactive_power**”: ”string”,
	”**power_factor**”: ”string”,
	”**dev_energy**”: ”string”,
	”**time**”: ”2022-11-22T18:24:19.289Z”
	}
	]

Tags:	datalist
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	Basic

5.2 Data

5.2.1 GET - datas

Table 5.3: **GET/ data/**

Request Format:	http(s)://mdw-api/v1/datas/
Operation Type:	GET
Operation Id:	datas_list
Description:	• voltage • current • active_power • reactive_power • power_factor • dev_energy • time
Parameters:	no-parametes
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-22T23:38:32.727Z”
	}
	]

Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.2.2 POST - datas/

Table 5.4: **POST/ datas/**

Request Format:	http(s)://mdw-api/v1/datas/
Operation Type:	POST
Operation Id:	datas_create
Description:	• voltage • current • active_power • reactive_power • power_factor • dev_energy • time
Parameters:	**name**:*data* **in**:*body* **required**:*true*
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	[
	{
	”**dev**”: 0,
	”**voltage**”: ”string”,
	”**current**”: ”string”,
	”**active_power**”: ”string”,
	”**reactive_power**”: ”string”,
	”**power_factor**”: ”string”,
	”**dev_energy**”: ”string”,
	”**time**”: ”2022-11-22T16:49:05.829Z”
	}
	]

Tags:	data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.2.3 GET - datas/ id/

Table 5.5: **GET/ datas/ id/**

Request Format:	http(s)://mdw-api/v1/datas/ id/
Operation Type:	GET
Operation Id:	datas_read
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-23T00:04:30.404Z”
	}

Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.2.4 PUT - datas/ id/

Table 5.6: **PUT/ datas/ id/**

Request Format:	http(s)://mdw-api/v1/datas/ id/
Operation Type:	PUT
Operation Id:	datas_update
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path) **data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-23T00:04:30.412Z”
	}

Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.2.5 PATCH - datas/ id/

Table 5.7: **PATCH/ datas/ id/**

Request Format:	http(s)://mdw-api/v1/datas/ id/
Operation Type:	PATCH
Operation Id:	datas_partial_update
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path) **data**: *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	{
	"dev": 0,
	"voltage": "string",
	"current": "string",
	"active_power": "string",
	"reactive_power": "string",
	"power_factor": "string",
	"dev_energy": "string",
	"time": "2022-11-22T23:13:33.448Z"
	}

Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.2.6 DELETE - datas/ id/

Table 5.8: **DELETE/ datas/ id/**

Request Format:	http(s)://mdw-api/v1/datas/ id/
Operation Type:	DELETE
Operation Id:	datas_delete
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3 Devices

5.3.1 GET - devices/

Table 5.9: **GET/ devices/**

Request Format:	http(s)://mdw-api/v1/devices
Operation Type:	GET
Operation Id:	devices_list
Description:	• devName • devType • dev_on
Parameters:	no-parametes
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”**id**”: 0,
	”**devName**”: ”string”,
	”**devType**”:”Generacao”,
	”**dev_on**”: ”true”,
	}
	]

Tags:	devices
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3.2 POST - devices/

Table 5.10: **POST/ devices/**

Request Format:	http(s)://mdw-api/v1/devices
Operation Type:	POST
Operation Id:	devices_create
Description:	• devName • devType • dev_on
Parameters:	**data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	{
	”id”: 0,
	”devName”: ”string”,
	”devType”:”Generacao”,
	”dev_on”: ”true”,
	}

Tags:	devices
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3.3 GET - devices/ id/

Table 5.11: **GET/ devices/ id/**

Request Format:	http(s)://mdw-api/v1/devices/id/
Operation Type:	GET
Operation Id:	devices_read
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”devName”: ”string”,
	”devType”:”Generacao”,
	”dev_on”: ”true”,
	}

Tags:	devices
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3.4 PUT - devices/ id/

Table 5.12: **PUT/ devices/ id/**

Request Format:	http(s)://mdw-api/v1/devices/ id/
Operation Type:	PUT
Operation Id:	devices_update
Description:	• devName • devType • dev_on
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”devName”: ”string”,
	”devType”:”Generacao”,
	”dev_on”: ”true”,
	}

Tags:	devices
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3.5 PATCH - devices/ id/

Table 5.13: **PATCH/ devices/ id/**

Request Format:	http(s)://mdw-api/v1/devices/id
Operation Type:	PATCH
Operation Id:	devices_partial_update
Description:	• devName • devType • dev_on
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”devName”: ”string”,
	”devType”:”Generacao”,
	”dev_on”: ”true”,
	}

Tags:	devices
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.3.6 DELETE - devices/ id/

Table 5.14: **DELETE/ devices/ id/**

Request Format:	http(s)://mdw-api/v1/devices/id
Operation Type:	DELETE
Operation Id:	devices_delete
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	datas
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4 Features

5.4.1 GET - features/

Table 5.15: **GET/ features/**

Request Format:	http(s)://mdw-api/v1/features/
Operation Type:	GET
Operation Id:	features_list
Description:	• broker_address • cloud_method • cloud_time • update • request_time • update_date • user • password • template
Parameters:	no-paremeters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”**id**”: 0,
	”**broker_address**”: ”string”,
	”**cloud_method**”:”string”,
	”**cloud_time**”: ”string”,
	”**update**”: ”string”,
	”**request_time**”: ”string”,
	”**update_date**”:”2022-11-23T01:50:21.942Z”,
	”**user**”: ”string”,
	”**password**”: ”string”,
	”**template**”: ”string”,
	}

Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4.2 POST - features/

Table 5.16: **POST/ features/**

Request Format:	http(s)://mdw-api/v1/features/
Operation Type:	POST
Operation Id:	features_create
Description:	• broker_address • cloud_method • cloud_time • update • request_time • user • password • template
Parameters:	**data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	{
	”id”: 0,
	”broker_address”: ”string”,
	”cloud_method”:”string”,
	”cloud_time”: ”string”,
	”update”: ”string”,
	”request_time”: ”string”,
	”update_date”:”2022-11-23T02:25:24.695Z”,
	”user”: ”string”,
	”password”: ”string”,
	”template”: ”string”,
	}

Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4.3 GET - features/ id/

Table 5.17: **GET/ features/ id/**

Request Format:	http(s)://mdw-api/v1/features/ id/
Operation Type:	GET
Operation Id:	features_read
Description:	A unique integer value identifying this feature.
Parameters:	**id***required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”broker_address”: ”string”,
	”cloud_method”:”string”,
	”cloud_time”: ”string”,
	”update”: ”string”,
	”request_time”: ”string”,
	”update_date”:”2022-11-23T02:25:24.695Z”,
	”user”: ”string”,
	”password”: ”string”,
	”template”: ”string”,
	}

Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4.4 PUT - features/ id/

Table 5.18: **PUT/ features/ id/**

Request Format:	http(s)://mdw-api/v1/features/ id/
Operation Type:	PUT
Operation Id:	features_update
Description:	A unique integer value identifying this feature.
Parameters:	**data** *required object (body) **id***required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”broker_address”: ”string”,
	”cloud_method”:”string”,
	”cloud_time”: ”string”,
	”update”: ”string”,
	”request_time”: ”string”,
	”update_date”:”2022-11-23T02:25:24.695Z”,
	”user”: ”string”,
	”password”: ”string”,
	”template”: ”string”,
	}

Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4.5 PATCH - features/ id/

Table 5.19: **PATCH /features/ id/**

Request Format:	http(s)://mdw-api/v1/features/ id/
Operation Type:	PATCH
Operation Id:	features_partial_update
Description:	A unique integer value identifying this feature.
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”broker_address”: ”string”,
	”cloud_method”:”string”,
	”cloud_time”: ”string”,
	”update”: ”string”,
	”request_time”: ”string”,
	”update_date”:”2022-11-23T02:25:24.695Z”,
	”user”: ”string”,
	”password”: ”string”,
	”template”: ”string”,
	}

Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.4.6 DELETE - features/ id/

Table 5.20: **DELETE /features/ id/**

Request Format:	http(s)://mdw-api/v1/features/ id/
Operation Type:	DELETE
Operation Id:	features_delete
Description:	A unique integer value identifying this data.
Parameters:	**id**: *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	features
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.5 Initial

5.5.1 GET - Initial/

Table 5.21: **GET/ Initial/**

Request Format:	http(s)://mdw-api/v1/Initial/
Operation Type:	GET
Operation Id:	initial_list
Description:	Information Registration API HEMS.
Parameters:	no - parameters
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	Initial
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.5.2 PUT - Initial/

Table 5.22: **PUT/ Initial/**

Request Format:	http(s)://mdw-api/v1/Initial/
Operation Type:	PUT
Operation Id:	initial_update
Description:	Information Registration API HEMS..
Parameters:	no - parameters
Response content type:	application/json
Example/Value of Return:	**Responses**: 200
Tags:	Initial
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.6 Interfaces

5.6.1 GET - interfaces/

Table 5.23: **GET/ interfaces/**

Request Format:	http(s)://mdw-api/v1/interfaces/
Operation Type:	GET
Operation Id:	interfaces_list
Description:	Zigbee Coordinator Interface Configuration API.
Parameters:	no - parameters
Response content type:	application/json
Example/Value of Return:	**Responses**: 200
Tags:	interfaces
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.6.2 PUT - interfaces/

Table 5.24: **PUT/ interfaces/**

Request Format:	http(s)://mdw-api/v1/interfaces/
Operation Type:	PUT
Operation Id:	initial_update
Description:	Zigbee Coordinator Interface Configuration API.
Parameters:	no - parameters
Response content type:	application/json
Example/Value of Return:	**Responses**: 200
Tags:	interfaces
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.7 Outlets

5.7.1 GET - outlets/

Table 5.25: **GET/ outlets/**

Request Format:	http(s)://mdw-api/v1/outlets/
Operation Type:	GET
Operation Id:	outlets_list
Description:	-
Parameters:	no - parameters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”id”: 0,
	”label”: ”string”,
	”outletType”: true,
	”connectedDevice”: ”string”,
	”outletZone”: ”string”,
	”mac_address”: ”string”,
	”data”: [
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-30T03:40:49.633Z”
	}
	]
	}
	]

Tags:	outlets
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.7.2 POST - outlets/

Table 5.26: **POST/ outlets/**

Request Format:	http(s)://mdw-api/v1/outlets/
Operation Type:	POST
Operation Id:	outlets_create
Description:	• label • outletType • mac_address
Parameters:	**data*** required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	type: array
	{
	”id”: 0,
	”label”: ”string”,
	”outletType”: true,
	”connectedDevice”: ”string”,
	”outletZone”: ”string”,
	”mac_address”: ”string”,
	”data”: [
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-30T03:42:03.417Z”
	}
	]
	}

Tags:	outlets
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.7.3 PUT - outlets/set_device/ id/

Table 5.27: **PUT/ outlets/set_device/ id/**

Request Format:	http(s)://mdw-api/v1/outlets/set_device/id/
Operation Type:	PUT
Operation Id:	outlets_set_device
Description:	• label. • outletType. • mac_address.
Parameters:	**data*** required object (body) **id*** required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”**id**”: 0,
	”**label**”: ”string”,
	”**outletType**”: true,
	”**connectedDevice**”: ”string”,
	”**outletZone**”: ”string”,
	”**mac_address**”: ”string”,
	”**data**”: [
	{
	”**dev**”: 0,
	”**voltage**”: ”string”,
	”**current**”: ”string”,
	”**active_power**”: ”string”,
	”**reactive_power**”: ”string”,
	”**power_factor**”: ”string”,
	”**dev_energy**”: ”string”,
	”**time**”: ”2022-11-30T03:43:54.620Z”
	}
	]
	}

Tags:	outlets
Restrictions:	-
Security Requirements:	basic

5.7.4 GET - outlets/ id/

Table 5.28: **GET/ outlets/ id/**

Request Format:	http(s)://mdw-api/v1/outlets/id/
Operation Type:	GET
Operation Id:	outlets_read
Description:	A unique integer value identifying this outlet.
Parameters:	**id*** required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”id”: 0,
	”label”: ”string”,
	”outletType”: true,
	”connectedDevice”: ”string”,
	”outletZone”: ”string”,
	”mac_address”: ”string”,
	”data”: [
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-30T03:47:21.693Z”
	}
	]
	}
	]

Tags:	outlets
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.7.5 PUT - outlets/ id/

Table 5.29: **PUT/ outlets/ id/**

Request Format:	http(s)://mdw-api/v1/outlets/id/
Operation Type:	PUT
Operation Id:	outlets_update
Description:	• label. • outletType. • mac_address.
Parameters:	**data*** required object (body) **id*** required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”**id**”: 0,
	”**label**”: ”string”,
	”**outletType**”: true,
	”**connectedDevice**”: ”string”,
	”**outletZone**”: ”string”,
	”**mac_address**”: ”string”,
	”**data**”: [
	{
	”**dev**”: 0,
	”**voltage**”: ”string”,
	”**current**”: ”string”,
	”**active_power**”: ”string”,
	”**reactive_power**”: ”string”,
	”**power_factor**”: ”string”,
	”**dev_energy**”: ”string”,
	”**time**”: ”2022-11-30T03:50:19.230Z”
	}
	]
	}

Tags:	outlets
Restrictions:	-
Security Requirements:	basic

5.7.6 PATCH - outlets/ id/

Table 5.30: **PATCH/ outlets/ id/**

Request Format:	http(s)://mdw-api/v1/outlets/id/
Operation Type:	PATCH
Operation Id:	outlets_partial_update
Description:	• label. • outletType • mac_address
Parameters:	**data***required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	"id": 0,
	"label": "string",
	"outletType": true,
	"connectedDevice": "string",
	"outletZone": "string",
	"mac_address": "string",
	"data": [
	{
	"dev": 0,
	"voltage": "string",
	"current": "string",
	"active_power": "string",
	"reactive_power": "string",
	"power_factor": "string",
	"dev_energy": "string",
	"time": "2022-11-30T03:51:47.160Z"
	}
	]
	}

Tags:	outlets
Restrictions:	-
Security Requirements:	basic

5.7.7 DELETE - outlets/ id/

Table 5.31: **DELETE/ outlets/ id/**

Request Format:	http(s)://mdw-api/v1/outlets/ id/
Operation Type:	DELETE
Operation Id:	outlets_delete
Description:	A unique integer value identifying this data.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	outlets
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.7.8 GET - outlets/ id/ data/

Table 5.32: **GET/ outlets/id/ data/**

Request Format:	http(s)://mdw-api/v1/outlets/ id/
Operation Type:	GET
Operation Id:	outlets_data
Description:	A unique integer value identifying this outlet.
Parameters:	**id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	{
	”id”: 0,
	”label”: ”string”,
	”outletType”: true,
	”connectedDevice”: ”string”,
	”outletZone”: ”string”,
	”mac_address”: ”string”,
	”data”: [
	{
	”dev”: 0,
	”voltage”: ”string”,
	”current”: ”string”,
	”active_power”: ”string”,
	”reactive_power”: ”string”,
	”power_factor”: ”string”,
	”dev_energy”: ”string”,
	”time”: ”2022-11-30T03:52:34.231Z”
	}
	]
	}

Tags:	outlets
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8 Plug Request

5.8.1 GET - plugRequest/

Table 5.33: **GET/ plugRequest/**

Request Format:	http(s)://mdw-api/v1/plugRequest/
Operation Type:	GET
Operation Id:	plugRequest_list
Description:	-
Parameters:	no - parameters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	
	[
	{
	”**id**”: 0,
	”**readRelayStatus**”: true,
	”**onOff**”: true,
	”**onOffTimer**”: 0,
	”**reset**”: true,
	”**switchedOutput**”: ”string”,
	”**plugId**”: ”string”
	}
	]

Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8.2 POST - plugRequest/

Table 5.34: **POST/ plugRequest/**

Request Format:	http(s)://mdw-api/v1/plugRequest/
Operation Type:	POST
Operation Id:	plugRequest_create
Description:	• readRelayStatus • onOff • onOffTimer • reset • switchedOutput • plugId
Parameters:	**id** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	type: array
	
	[
	{
	”**id**”: 0,
	”**readRelayStatus**”: true,
	”**onOff**”: true,
	”**onOffTimer**”: 0,
	”**reset**”: true,
	”**switchedOutput**”: ”string”,
	”**plugId**”: ”string”
	}
	]

Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8.3 GET - plugRequest/ id/

Table 5.35: **GET/ plugRequest/ id/**

Request Format:	http(s)://mdw-api/v1/plugRequest/ id/
Operation Type:	GET
Operation Id:	plugRequest_read
Description:	• readRelayStatus • onOff • onOffTimer • reset • switchedOutput • plugId
Parameters:	A unique integer value identifying this plugRequest.
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	[
	{
	”**id**”: 0,
	”**readRelayStatus**”: true,
	”**onOff**”: true,
	”**onOffTimer**”: 0,
	”**reset**”: true,
	”**switchedOutput**”: ”string”,
	”**plugId**”: ”string”
	}
	]

Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8.4 PUT - plugRequest/ id/

Table 5.36: **PUT/ plugRequest/ id/**

Request Format:	http(s)://mdw-api/v1/plugRequest/ id/
Operation Type:	PUT
Operation Id:	plugRequest_update
Description:	**data** *required object (body) **id** *required integer (path)
Parameters:	A unique integer value identifying this plugRequest.
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	[
	{
	”**id**”: 0,
	”**readRelayStatus**”: true,
	”**onOff**”: true,
	”**onOffTimer**”: 0,
	”**reset**”: true,
	”**switchedOutput**”: ”string”,
	”**plugId**”: ”string”
	}
	]

Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8.5 PATCH - plugRequest/ id/

Table 5.37: **PATCH/ plugRequest/ id/**

Request Format:	http(s)://mdw-api/v1/plugRequest/ id/
Operation Type:	PATCH
Operation Id:	plugRequest_partial_update
Description:	**data** *required object (body) **id** *required integer (path)
Parameters:	A unique integer value identifying this plugRequest.
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	[
	{
	”id”: 0,
	”readRelayStatus”: true,
	”onOff”: true,
	”onOffTimer”: 0,
	”reset”: true,
	”switchedOutput”: ”string”,
	”plugId”: ”string”
	}
	]

Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.8.6 DELETE - plugRequest/ id/

Table 5.38: **DELETE/ plugRequest/ id/**

Request Format:	http(s)://mdw-api/v1/plugRequest/id/
Operation Type:	DELETE
Operation Id:	plugRequest_delete
Description:	A unique integer value identifying this plugRequest id.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	plugRequest
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.9 Plug Response

5.9.1 GET - plugResponse/

Table 5.39: **GET**/ **plugResponse**/

Request Format:	http(s)://mdw-api/v1/plugResponse/
Operation Type:	GET
Operation Id:	plugResponse_list
Description:	• responseId • relayStatus • onOffStatus • switchedOutputStatus
Parameters:	no - parameters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”**responseId**”: 0,
	”**relayStatus**”: true,
	”**onOffStatus**”: true,
	”**switchedOutputStatus**”: true,
	}
	]

Tags:	plugResponse
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.9.2 POST - plugResponse/

Table 5.40: **POST/ plugResponse/**

Request Format:	http(s)://mdw-api/v1/plugResponse/
Operation Type:	POST
Operation Id:	plugResponse_create
Description:	• responseId • relayStatus • onOffStatus • switchedOutputStatus
Parameters:	**data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	type: array
	
	{
	”**responseId**”: 0,
	”**relayStatus**”: true,
	”**onOffStatus**”: true,
	”**switchedOutputStatus**”: true,
	}

Tags:	plugResponse
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.9.3 GET - plugResponse/ id/

Table 5.41: **GET/ plugResponse/ id/**

Request Format:	http(s)://mdw-api/v1/plugResponse/id
Operation Type:	GET
Operation Id:	plugResponse_read
Description:	A unique integer value identifying this plugResponses
Parameters:	**id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	{
	”**responseId**”: 0,
	”**relayStatus**”: true,
	”**onOffStatus**”: true,
	”**switchedOutputStatus**”: true,
	}

Tags:	plugResponse
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.9.4 PATCH - plugResponse/ id/

Table 5.42: **PATCH/ plugResponse/ id/**

Request Format:	http(s)://mdw-api/v1/plugResponse/id
Operation Type:	PATCH
Operation Id:	plugResponse_partial_update
Description:	A unique integer value identifying this plugResponses id.
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	
	{
	”**responseId**”: 0,
	”**relayStatus**”: true,
	”**onOffStatus**”: true,
	”**switchedOutputStatus**”: true,
	}

Tags:	plugResponse
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.9.5 DELETE - plugResponse/ id/

Table 5.43: **DELETE/ plugResponse/ id/**

Request Format:	http(s)://mdw-api/v1/ plugResponse/ id/
Operation Type:	DELETE
Operation Id:	plugResponse_delete
Description:	A unique integer value identifying this plugRequest id.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	plugResponse
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10 Root Data

5.10.1 GET - root-data/

Table 5.44: **GET/ root-data/**

Request Format:	http(s)://mdw-api/v1/root-data/
Operation Type:	GET
Operation Id:	root-data_list
Description:	• time • power
Parameters:	no - parameters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”**time**”:”2022-11-23T08:40:42.882Z”,
	”**power**: ”string”,
	}
	]

Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10.2 POST - root_data/

Table 5.45: **POST/ root-data/**

Request Format:	http(s)://mdw-api/v1/root-data/
Operation Type:	POST
Operation Id:	root-data_create
Description:	• time • power
Parameters:	**data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	type: array
	[
	{
	”**time**”:”2022-11-23T08:46:16.022Z”,
	”**power**: ”string”,
	}
	]

Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10.3 GET - root_data/ id/

Table 5.46: **GET /root-data/ id/**

Request Format:	http(s)://mdw-api/v1/root-data/
Operation Type:	GET
Operation Id:	root-data_read
Description:	A unique integer value identifying this root data
Parameters:	**data** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”time”:”2022-11-23T08:52:17.906Z”,
	”power: ”string”,
	}
	]

Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10.4 PUT - root_data/ id/

Table 5.47: **PUT/ root-data/ id/**

Request Format:	http(s)://mdw-api/v1/root-data/ id/
Operation Type:	PUT
Operation Id:	root-data_update
Description:	A unique integer value identifying this root data id • Time • Power
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	[
	{
	”**time**”:”2022-11-23T09:12:13.161Z”,
	”**power**: ”string”,
	}
	]

Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10.5 PATCH - root_data/ id/

Table 5.48: **PATCH/ root-data/ id/**

Request Format:	http(s)://mdw-api/v1/root-data/id
Operation Type:	PATCH
Operation Id:	root-data_partial_update
Description:	A unique integer value identifying this root data id • Time • Power
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	[
	{
	”**time**”:”2022-11-23T09:15:14.894Z”,
	”**power**: ”string”,
	}
	]

Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.10.6 DELETE - root_data/ id/

Table 5.49: **DELETE/ root-data/ id/**

Request Format:	http(s)://mdw-api/v1/ root-data/ id/
Operation Type:	DELETE
Operation Id:	root-data_delete
Description:	A unique integer value identifying this root data id.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	root-data
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11 Systems Hems

5.11.1 GET - systems_hems/

Table 5.50: **GET/ systems_hems/**

Request Format:	http(s)://mdw-api/v1/systems_hems/
Operation Type:	GET
Operation Id:	systems_hems_list
Description:	• hems_last_update • userName • homeCity • homeStreet • homeNeighbour • homeComplement • precoKWh
Parameters:	no - parameters
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	type: array
	[
	{
	”id”:0,
	”hems_last_update”: ”2022-11-23T10:58:51.957Z”,
	”userName”: ”string”,
	”homeCity”: ”string”,
	”homeStreet”: ”string”,
	”homeNeighbour”: ”string”,
	”homeComplement”: ”string”,
	”precoKWh”: ”string”,
	}
	]

Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11.2 POST - systems_hems/

Table 5.51: **POST/ systems_hems/**

Request Format:	http(s)://mdw-api/v1/systems_hems/
Operation Type:	POST
Operation Id:	systems_hems_create
Description:	• hems_last_update • userName • homeCity • homeStreet • homeNeighbour • homeComplement • precoKWh
Parameters:	**data** *required object (body)
Response content type:	application/json

Example/Value of Return:	**Responses**: 201
	[
	{
	”**id**”:0,
	”**hems_last_update**”: ”2022-11-23T10:58:51.961Z”,
	”**userName**”: ”string”,
	”**homeCity**”: ”string”,
	”**homeStreet**”: ”string”,
	”**homeNeighbour**”: ”string”,
	”**homeComplement**”: ”string”,
	”**precoKWh**”: ”string”,
	}
	]

Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11.3 GET - systems_hems/ id/

Table 5.52: **POST/ systems_hems/ id/**

Request Format:	http(s)://mdw-api/v1/systems_hems/ id/
Operation Type:	GET
Operation Id:	systems_hems_read
Description:	A unique integer value identifying this hems_sys id.
Parameters:	**id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	[
	{
	”id”:0,
	”hems_last_update”: ”2022-11-23T11:11:19.865Z”,
	”userName”: ”string”,
	”homeCity”: ”string”,
	”homeStreet”: ”string”,
	”homeNeighbour”: ”string”,
	”homeComplement”: ”string”,
	”precoKWh”: ”string”,
	}
	]

Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11.4 PUT - systems_hems/ id/

Table 5.53: **PUT/ systems_hems/ id/**

Request Format:	http(s)://mdw-api/v1/systems_hems/ id/
Operation Type:	PUT
Operation Id:	systems_hems_update
Description:	A unique integer value identifying this hems_sys id.
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	[
	{
	”**id**”:0,
	”**hems_last_update**”: ”2022-11-23T11:16:45.028Z”,
	”**userName**”: ”string”,
	”**homeCity**”: ”string”,
	”**homeStreet**”: ”string”,
	”**homeNeighbour**”: ”string”,
	”**homeComplement**”: ”string”,
	”**precoKWh**”: ”string”,
	}
	]

Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11.5 PATCH - systems_hems/ id/

Table 5.54: **PATCH/ systems_hems/ id/**

Request Format:	http(s)://mdw-api/v1/systems_hems/ id/
Operation Type:	PATCH
Operation Id:	systems_hems_partial_update
Description:	A unique integer value identifying this hems_sys id.
Parameters:	**data** *required object (body) **id** *required integer (path)
Response content type:	application/json

Example/Value of Return:	**Responses**: 200
	
	[
	{
	”**id**”:0,
	”**hems_last_update**”: ”2022-11-23T11:22:17.391Z”,
	”**userName**”: ”string”,
	”**homeCity**”: ”string”,
	”**homeStreet**”: ”string”,
	”**homeNeighbour**”: ”string”,
	”**homeComplement**”: ”string”,
	”**precoKWh**”: ”string”,
	}
	]

Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

5.11.6 DELETE/ systems_hems/ id/

Table 5.55: **DELETE - systems_hems/ id/**

Request Format:	http(s)://mdw-api/v1/ systems_hems/ id/
Operation Type:	DELETE
Operation Id:	systems_hems_delete
Description:	A unique integer value identifying this hems_sys id.
Parameters:	**id** *required integer (path)
Response content type:	application/json
Example/Value of Return:	**Responses**: 204
Tags:	systems_hems
Restrictions:	This API should only be accessible to non-local clients and local stand-alone clients
Security Requirements:	basic

6 Description of models REST APIs

6.1 Model Data

Table 6.1: **Model Data**

Name	**Description**
Data required:	• Voltage • Current • Active Power • Reactive Power • Power Factor • Dev Energy • Time
Dev:	**type:** integer **x-nullable:** true
Voltage:	**type:** string **maxLength:** 255 **minLength:** 1
Current:	**type:** string **maxLength:** 255 **minLength:** 1
Active Power:	**type:** string **maxLength:** 255 **minLength:** 1
Reactive Power:	**type:** string **maxLength:** 255 **minLength:** 1
Power Factor:	**type:** string **maxLength:** 255 **minLength:** 1
Dev Energy:	**type:** string **maxLength:** 6 **minLength:** 1
Active Power:	**type:** string **format:** data-time

6.2 Model Devices

Table 6.2: **Model Devices**

Name	**Description**
Devices required:	• devName • devType
Type:	**title:** object
Id:	**title:** ID **type:** integer **readOnly:** true
devName:	**title:** DevName **type:** string **maxLength:** 30 **minLength:** 1
devType:	**title:** devType **type:** string
enum:	• Generation • Consumo
dev_on:	**title:** dev_on **type:** boolean

6.3 Model Features

Table 6.3: **Model Features**

Name	Description
broker_address:	**title:** broker_address **type:** string **maxLength:** 255 **minLength:** 1
cloud method:	**title:** Cloud_method **type:** string **maxLength:** 30 **minLength:** 1
cloud time:	**title:** cloud_time **type:** string **maxLength:** 10 **minLength:** 1
update:	**title:** Update **type:** string **format** uri **maxLength:** 255 **minLength:** 1
request time:	**title:** Request_time **type:** string **maxLength:** 10 **minLength:** 1
update date:	**title:** Update_date **type:** string **format:** date-time **readOnly:** true
user:	**title:** User **type:** string **maxLength:** 50 **minLength:** 1
password:	**title:** Password **type:** string **maxLength:** 50 **minLength:** 1
template:	**title:** Template **type:** string **maxLength:** 255 **minLength:** 1

6.4 Model Outlet

Table 6.4: **Model Outlet**

Name	**Description**
Id:	**title** ID **type:** integer **readOnly:** true
label:	**title:** Label **type:** string **maxLength:** 50 **x-nullable:** true
outletType:	**title:** OutletType **type:** boolean **x-nullable:** true
connectedDevice:	**title:** ConnectedDevice **type:** string **readOnly** true
outletZone:	**title:** OutletZone **type:** string **readOnly** true
mac address:	**title:** mac_address **type:** string **maxLength:** 50 **x-nullable:** true
data:	**type:** array **items:** • Current • Active Power • Reactive Power • Power Factor • Dev Energy • Time **readOnly:** true

6.5 Model Plug Request

Table 6.5: **Model Plug Request**

Name	**Description**
Plug Request Required:	**title**: plugId
Id:	**title:** ID **type:** integer **x-nullable:** true **readOnly:** true
Read Relay Status:	**title:** ReadRelayStatus **type:** boolean **x-nullable:** true
onOff:	**title:** onOff **type:** boolean **x-nullable:** true
onOffTimer:	**title:** onOffTimer **type:** integer **x-nullable:** true
reset:	**title:** reset **type:** boolean **x-nullable:** true
switchedOutput:	**title:** switchedOutput **type:** string **maxLength:** 255 **minLength:** 1 **x-nullable:** true
plugId:	**title:** plugId **type:** string **maxLength:** 255 **minLength:** 1

6.6 Model Plug Response

Table 6.6: **Model Plug Response**

Name	Description
Plug Response Required:	• responseId • relayStatus • onOffStatus • switchedOutputStatus
ResponseId:	**title:** responseId **type:** integer
RelayStatus:	**title:** relayStatus **type:** boolean
OnOffStatus:	**title:** OnOffStatus **type:** boolean
SwitchedOutputStatus:	**title:** SwitchedOutputStatus **type:** boolean

6.7 Model Root Data

Table 6.7: **Model Root Data**

Name	Description
RootData required:	• Time • Power
Time:	**title:** Time **type:** string **format:** data-time
Power:	**title:** power **type:** string **maxLength:** 50 **minLength:** 1

6.8 Model Hems_Sys

Table 6.8: **Model Hems_Sys**

Name	Description
Hems_sys required:	• hems_last_update • precoKWh
Id:	**title:** ID **type:** integer **readOnly:** true
hems_last_update:	**title:** hems_last_update **type:** string
userName:	**title:** userName **type:** string **maxLength:** 30 **minLength:** 1
homeCity:	**title:** HomeCity **type:** string **maxLength:** 30 **minLength:** 1
homeStreet:	**title:** HomeStreet **type:** string **maxLength:** 30 **minLength:** 1
homeNeighbour:	**title:** HomeNeighbour **type:** string **maxLength:** 20 **minLength:** 1
homeComplement:	**title:** HomeComplement **type:** string **maxLength:** 7 **minLength:** 1
precoKWh:	**title:** Template **type:** string **format:** decimal

7 Json Format REST API Codes

7.1 API data

```
Data:
   required:
     - voltage
     - current
     - active_power
     - reactive_power
     - power_factor
     - dev_energy
     - time
   type: object
   properties:
     dev:
       title: Dev
       type: integer
       x-nullable: true
     voltage:
       title: Voltage
       type: string
       maxLength: 255
       minLength: 1
     current:
       title: Current
       type: string
       maxLength: 255
       minLength: 1
     active_power:
       title: Active power
       type: string
       maxLength: 255
       minLength: 1
     reactive_power:
       title: Reactive power
       type: string
```

```
        maxLength: 255
        minLength: 1
      power_factor:
        title: Power factor
        type: string
        maxLength: 255
        minLength: 1
      dev_energy:
        title: Dev energy
        type: string
        maxLength: 6
        minLength: 1
      time:
        title: Time
        type: string
        format: date-time
```

7.2 API Device

```
Device:
    required:
      - devName
      - devType
    type: object
    properties:
      id:
        title: ID
        type: integer
        readOnly: true
      devName:
        title: DevName
        type: string
        maxLength: 30
        minLength: 1
      devType:
        title: DevType
        type: string
        enum:
          - Generacao
          - Consumo
      dev_on:
        title: Dev on
        type: boolean
```

7.3 API Feature

```
Feature:
    required:
      - broker_address
      - cloud_method
      - cloud_time
      - update
      - request_time
      - user
      - password
      - template
    type: object
    properties:
      id:
        title: ID
        type: integer
        readOnly: true
      broker_address:
        title: Broker address
        type: string
        maxLength: 255
        minLength: 1
      cloud_method:
        title: Cloud method
        type: string
        maxLength: 30
        minLength: 1
      cloud_time:
        title: Cloud time
        type: string
        maxLength: 10
        minLength: 1
      update:
        title: Update
        type: string
        format: uri
        maxLength: 255
        minLength: 1
      request_time:
        title: Request time
        type: string
        maxLength: 10
        minLength: 1
      update_date:
        title: Update date
        type: string
        format: date-time
        readOnly: true
      user:
        title: User
        type: string
        maxLength: 50
        minLength: 1
      password:
        title: Password
        type: string
        maxLength: 50
        minLength: 1
      template:
        title: Template
        type: string
        maxLength: 255
        minLength: 1
```

7.4 API Outlet

```
Outlet:
    type: object
    properties:
      id:
        title: ID
        type: integer
        readOnly: true
      label:
        title: Label
        type: string
        maxLength: 50
        x-nullable: true
      outletType:
        title: OutletType
        type: boolean
        x-nullable: true
      connectedDevice:
        title: Connecteddevice
        type: string
        readOnly: true
      outletZone:
        title: Outletzone
        type: string
        readOnly: true
      mac_address:
        title: Mac address
        type: string
        maxLength: 50
        x-nullable: true
      data:
        type: array
        items:
          $ref: ’#/definitions/Data’
        readOnly: true
```

7.5 API PlugRequest

```
PlugRequest:
    required:
      - plugId
    type: object
    properties:
      id:
        title: ID
        type: integer
        readOnly: true
      readRelayStatus:
        title: ReadRelayStatus
        type: boolean
        x-nullable: true
      onOff:
        title: OnOff
        type: boolean
        x-nullable: true
      onOffTimer:
        title: OnOffTimer
```

```
       type: integer
       x-nullable: true
     reset:
       title: Reset
       type: boolean
       x-nullable: true
     switchedOutput:
       title: SwitchedOutput
       type: string
       maxLength: 255
       minLength: 1
       x-nullable: true
     plugId:
       title: PlugId
       type: string
       maxLength: 255
       minLength: 1
```

7.6 API PlugResponse

```
PlugResponse:
   required:
     - responseId
     - relayStatus
     - onOffStatus
     - switchedOutputStatus
   type: object
   properties:
     responseId:
       title: ResponseId
       type: integer
     relayStatus:
       title: RelayStatus
       type: boolean
     onOffStatus:
       title: OnOffStatus
       type: boolean
     switchedOutputStatus:
       title: SwitchedOutputStatus
       type: boolean
```

7.7 API RootData

```
RootData:
    required:
      - time
      - power
    type: object
    properties:
      time:
        title: Time
        type: string
        format: date-time
      power:
        title: Power
        type: string
        maxLength: 50
        minLength: 1
```

7.8 API Hems_sys

```
 Hems_sys:
    required:
      - hems_last_update
      - precoKWh
    type: object
    properties:
      id:
        title: ID
        type: integer
        readOnly: true
      hems_last_update:
        title: Hems last update
        type: string
        format: date-time
      userName:
        title: UserName
        type: string
        maxLength: 30
        minLength: 1
      homeCity:
        title: HomeCity
        type: string
        maxLength: 30
        minLength: 1
      homeStreet:
        title: HomeStreet
        type: string
        maxLength: 30
        minLength: 1
      homeNeighbour:
        title: HomeNeighbour
        type: string
        maxLength: 20
        minLength: 1
      homeComplement:
        title: HomeComplement
        type: string
        maxLength: 7
        minLength: 1
      precoKWh:
        title: PrecoKWh
        type: string
        format: decimal
```

8 References

[1] Bray, T. (Ed.), “The JavaScript Object Notation (JSON) Data Interchange Format”, RFC 8259, DOI 10.17487/RFC8259 , Dezembro de 2017, disponível em http://www.rfc-editor.org/info/rfc8259

[2] Fielding, R. (Ed.) e Reschke, J. (Ed.), “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing”, RFC 7230, DOI 10.17487/RFC7230, Junho de 2014, disponível em http://www.rfc-editor.org/info/rfc7230

[3] Paul C., Jamshidi P., Microservices: A systematic Mapping Study, Proceedings of the 6th International Conference on Cloud Computing and Services Science — Volume 1 and 2, (CLOSER 2016).

[4] OpenAPI Specification V3.0.2, https://github.com/RT-DSP/SHArM/blob/main/APIs-Documentation.pdf

[5] MQTT V3.1 Protocol Specification, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html