TECHPARK SÜDTIROL / ALTO ADIGE

OCPI PROFILE FOR E-MOBILITY DATA SHARING WITH THE OPEN DATA HUB

Sharing electric vehicles e-charging stations data in real-time with the Open Data Hub platform through the open standard OCPI

NOI AG / S.p.A.
A.-Volta-Straße 13A
Via A. Volta, 13A
I-39100 Bozen / Bolzano
T +39 0471 066 600
info@noi.bz.it
www.noi.bz.it

Nature of Innovation.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

DOCUMENT HISTORY

DATE	VERSION	AUTHOR	DESCRIPTION
08/03/2023	v1.0	Roberto Cavaliere	First document version draft

Introduction

This document aims to introduce a new standardized way in which e-charging data are shared with the Open Data Hub managed by NOI.

At present, various CPOs (i.e. Neogy, route220, Driwe) share their real-time data with the Open Data Hub according to a custom interface that was jointly defined in 2017. At that time, no standardized protocols for the sharing of e-mobility data were available or in a mature phase for the adoption.

In the last five years the diffusion of e-mobility and interoperability among the CPOs present on the market significantly pushed forward the definition and deployment of open standard protocols.

The reference open standard that is going to be used is the **OCPI protocol**, in its current version 2.2.1[1]. OCPI was introduced in the Netherlands and has become one of the reference standards in this specific domain. OCPI also supports the specific use case that we foresee in South Tyrol, i.e. a network of different CPOs sharing their data with a hub which can make them at disposal to eMSPs ore more in general to interested Data Consumers.

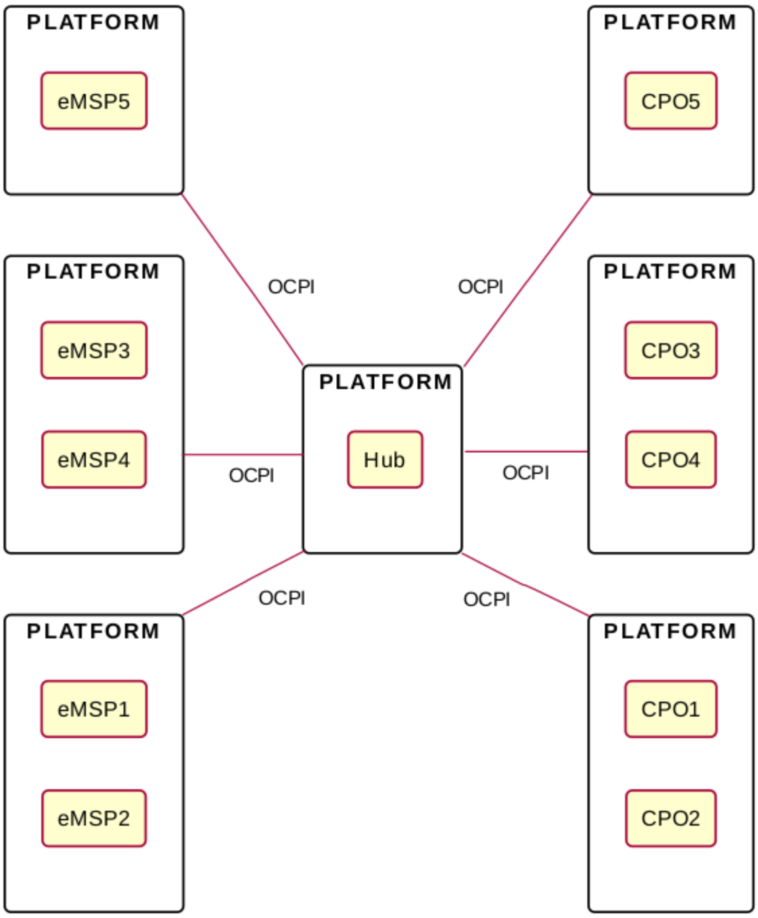

Figure 1: Platforms connected via a Hub topology example [1].

[1] https://evroaming.org/downloads/

Scope of the document

This document aims to specify the implementation aspects of the OCPI specification for the South Tyrolean use case.

In the gray formatted boxes as this, specific South Tyrolean design choices on how to implement the OCPI protocol are highlighted.

In this first implementation, the Open Data Hub aims only at receiving the real-time information of the status of the charging stations, by using the “**Location**” module. Further use cases that are possible thanks to the OCPI protocol are currently not considered but could be evaluated in future implementations.

Glossary

Abbreviations

Table 1: List of abbreviations.

ABBREVIATION	DESCRIPTION
CDR	Charge Detail Record
CPO	Charge Point Operator
eMSP	e-Mobility Service Provider
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
OCPI	Open Charge Point Interface
OCPP	Open Charge Point Protocol

Cardinality convention

The following cardinality convention is used:

- “**?**”: optional object
- “**1**”: mandatory object
- “*****”: a list of zero or more objects
- “**+**”: a list of at least one objects

Charging topology

The OCPI interface refers to the charging topology presented in Figure 2. Three entities are defined:

- “**Connector**”: is a specific socket or cable that is made available to an EV
- “**EVSE**”: is the part that controls the power supply to a single EV in a single session. An EVSE can have multiple connectors, but only one connector at a time can be active.
- “**Location**”: is a group of EVSE that belong to the same charging point.

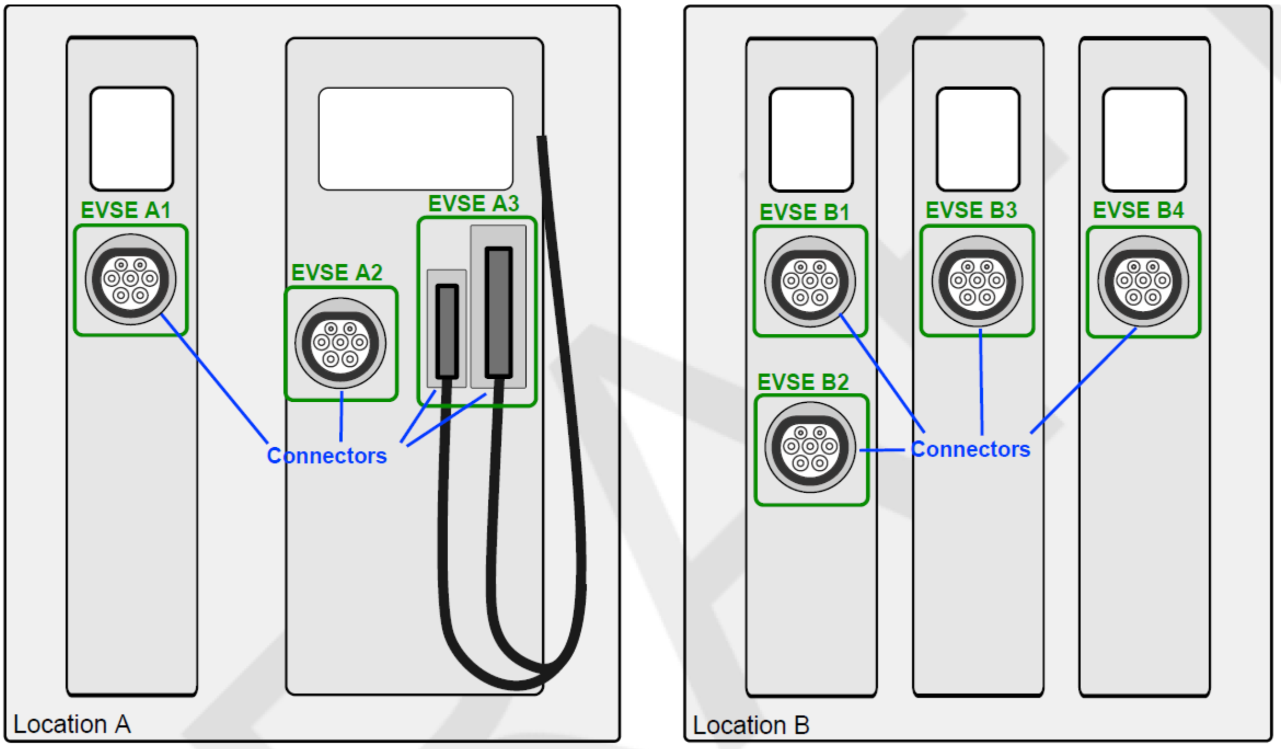

Figure 2: Charging topology graphical representation [1].

Transport and format

Communication protocol

The OCPI protocol is based on HTTP and uses the JSON format. A RESTful architecture of webservices is used.

Authentication

The OCPI protocol requests as mandatory the usage of SSL and token based authentication at HTTP level. More specifically, each OCPT HTTP request must add an “authorization” header, as for example

```
Authorization: Token IpbJOXxkxOAuKR92z0nEcmVF3Qw09VG7I7d/WCg0koM=
```

The token is called “**credentials token**” and is exchanged through the credentials module (see chapter “Credentials module”). The credentials token aims to uniquely identify the requesting party.

Communication modes

The OCPI protocol supports both “pull” and “push” models. The “push” model is optional, even if recommended.

In the South Tyrolean implementation, also based on the previous experience carried out so far, it is decided to support just the **pull mode**.

Communication methods

The OCPI protocol supports HTTP GET, PUT, PATCH or DELETE request methods.

In the South Tyrolean implementation, the **HTTP GET** request method is going to be used, unless there are particular use cases, which will be duly highlighted.

Pagination

The OCPI protocol foresees for the HTTP GET requests the support of pagination, in order to control the amount of objects exchanged. Pagination is enabled by additional URL parameters in the GET request. The parameters to be supported are listed in Table 2.

Table 2: Pagination request parameters.

PARAMETER	DESCRIPTION
offset	Indicates the starting object of the list to be returned. Default value is 0 (first object of the list)
limit	Indicates the number of objects that are requested. E.g. limit = 10 indicates a request of 10 objects.

It is important to underline that the server may decide by its own to return a reduced number of objects, despite the type of request received. In order to handle this, specific HTTP headers have to be added to any paginated GET response (see Table 3).

Table 3: Pagination HTTP header response.

HTTP HEADER	DESCRIPTION
Link	Link to the "next" page (if the current request does not refer to the last page)
X-Total-Count	Indicates the number of total objects available in relation to the received request, but excluding "limit" and "offset"
X-Limit	Indicates the number of objects returned

In the South Tyrolean implementation, the usage of **pagination** is considered **optional**. When used, it must follow the specification and the parameters foreseen.

Response format

The response message contains a JSON object with the properties presented in Table 4 .

Table 4: Response format properties.

PROPERTY	TYPE	CARDINALITY	DESCRIPTION
data	array / Object / String	* or ?	Response data object
status_code	int	1	Status code (see Status codes)
status_message	string	?	Status message (used for debugging)
timestamp	DateTime	1	Timestamp of message generation

Status codes

The OCPI protocol foresees two type of status codes: the standard HTTP codes and additional content-related codes. The latter ones have to be used in case the OCPI layer is additionally reached. The codes to be used are indicated in the tables below.

The **4xxx OCPI status codes (hub errors)** are **not considered** in the South Tyrolean implementation.

Table 5: HTTP status codes.

HTTP STATUS CODE	DESCRIPTION
400	Bad request (not valid JSON string)
404	Not found (no resource available)
401	Unauthorized (wrong token)
200	OK

Table 6: OCPI status codes.

OCPI STATUS CODE	DESCRIPTION
1000	Success code
2000	Invalid / missing parameters
2001	Not enough information
2002	Unknown location
3000	Generic server error
3001	Unable to use client's API (e.g. credentials registration)
3002	Unsupported version
3003	No matching end-points (during registration process)

Versions module

This is the required base module of OCPI. This module is the starting point for any OCPI connection. Via this module, clients can learn which versions of OCPI a server supports, and which modules it supports for each of the versions.

Request specification

HTTP GET requests to this module does not require any input parameter.

Response specification

The version module returns a list of version elements, which are characterized by the following fields.

Table 7: Version module – response specification.

PARAMETER	DESCRIPTION
version	The version number, e.g. “2.2.1”.
url	The end-point through which the data is made available through the indicated version

Example

```
[
   {
      "version": "2.1.1",
      "url": "https://www.server.com/ocpi/2.1.1/"
   },
   {
      "version": "2.2",
      "url": "https://www.server.com/ocpi/2.2/"
   }
]
```

Credentials module

The credentials module is used to exchange the credentials token that has to be used by parties for authorization of requests. Every OCPI request is required to contain a credentials token in the HTTP Authorization header.

The credentials module can be used in combination with the different HTTP methods, as summarized in Table 8.

Table 8: Credentials module and associated HTTP methods usage.

PARAMETER	DESCRIPTION
GET	Retrieves the credentials object to access the server's platform.
POST	Provides the server with a credentials object to access the client's system (i.e. register).
PUT	Provides the server with an updated credentials object to access the client's system.
DELETE	Informs the server that its credentials to the client's system are now invalid (i.e. unregister).

To start the exchange of credentials tokens, one platform has to be selected as Sender for the Credentials module. This has to be decided between the Platforms (outside of OCPI) before they first connect.

In the South Tyrolean implementation, the Open Data Hub is considered as the Sender, while the different charging stations data providers are considered as the Receiver.

To start the credentials exchange, the Receiver Platform must create a unique credentials token: CREDENTIALS_TOKEN_A that has to be used to authorize the Sender until the credentials exchange is finished. These credentials token along with the versions endpoint SHOULD be sent to the Sender in a secure way that is outside the scope of this protocol.

The Sender starts the registration process, retrieves the version information and details (using CREDENTIALS_TOKEN_A in the HTTP Authorization header). The Sender generates a unique credentials token: CREDENTIALS_TOKEN_B, sends it to the Receiver in a POST request to the credentials module of the Receiver. The Receiver stores CREDENTIALS_TOKEN_B and uses it for any requests to the Sender Platform, including the version information and details. The Receiver generates a unique credentials token: CREDENTIALS_TOKEN_C and returns it to the Sender in the response to the POST request from the Sender.

After the credentials exchange has finished, the Sender SHALL use CREDENTIALS_TOKEN_C in future OCPI request to the Receiver Platform. The CREDENTIALS_TOKEN_A can then be thrown away, it MAY no longer be used.

For all implementation details of the credentials module, please refer to the OCPI v2.2.1 specification document, Chapter 7.

Locations module

The Locations module is the key module in this OCPI implementation since it allows the sharing of location data and status information. The Location module can support both push (HTTP PUT) and pull mechanism (HTTP GET) for a receiver to receive the data.

In the South Tyrolean implementation, only the pull mechanism (HTTP GET) is foreseen.

OCPI standard does not recommend the sharing of locations that are not available for either public charging or roaming.

In the South Tyrolean implementation, however, we foresee the possibility for all access type charging stations to be exchanged through this module, since one of the use cases is to make statistical assessments about the growth of the charging infrastructures for electric vehicles in South Tyrol.

The reference charging topology is the one described in paragraph in Figure 2. API requests should have the following structure

Request list of locations

With this kind of request the information of a list of available locations, with EVSEs and connectors, can be obtained. The structure of the API request is:

```
{locations_endpoint_url}?[date_from={date_from}]&[date_to={date_to}]&[offset={off-
set}]&[limit={limit}]
```

All these parameters have to be considered as optional. If not given, all available charging stations have to be provided in the response, without limitations in the provided data.

Table 9: Locations module – request Location List parameters.

PARAMETER	DESCRIPTION
date_from	Only return Locations that have last_updated after or equal to this Date/Time (inclusive).
date_to	Only return Locations that have last_updated up to this Date/Time, but not including (exclusive).
offset	The offset of the first object returned. Default is 0.
limit	Maximum number of objects to GET.

The expected result is a list of Location objects. The following fields describing a single Location must be supported.

Table 10: Locations module – response Location List parameters.

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
country_code	String	1	ISO-3166 alpha-2 country code of the CPO that 'owns' this Location (length of the string = 2 characters)
party_id	String	1	ID of the CPO that 'owns' this Location (following the ISO-15118 standard). Length of the string = 3 characters
id	String	1	ID of the Location with the CPO platform (maximum length of the string = 36 characters)
publish	Boolean	1	Defines if a Location may be published. **Should be always set as TRUE.**
publish_allowed_to	Structure	*	This field may only be used when the publish field is set to false. **Not to be used.**

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
name	String	**1**	Display name of the location (maximum length of the string = 255 characters)
address	String	1	Street/block name and house number if available (maximum length of the string = 45 characters)
city	String	1	City or town (maximum length of the string = 45 characters)
postal_code	String	?	Postal code of the location (maximum length of the string = 10 characters)
state	String	?	State or province of the location (maximum length of the string = 20 characters)
country	String	1	ISO 3166-1 alpha-3 code for the country of this location.
coordinates	Structure	1	Coordinates of the location, expressed in the WGS 84 format in decimal degree. The structure contains the following fields: • Latitude • Longitude
related_locations	Structure	*	Geographical location of related points relevant to the user. **Not to be used.**
parking_type	Enum	*	The general type of parking at the charge point location. Admitted values: • ALONG_MOTORWAY • PARKING_GARAGE • PARKING_LOT • ON_DRIVEWAY • ON_STREET • UNDERGROUND_GARAGE
evses	Structure	**+**	List of EVSEs that belong to this Location. See Table 11
directions	Structure	*	Human-readable directions on how to reach the location. The structure contains the following fields: • language (language Code ISO 639-1) • text (maximum length = 512 characters)
operator	Structure	?	Information of the operator. The structure contains the following fields: • name (maximum length = 100 characters) • website (URL) • logo (image)
suboperator	Structure	?	Information of the sub-operator. The structure is the same as for operator.
owner	Structure	?	Information of the owner. The structure is the same as for operator.
facilities	Enum	*	Optional list of facilities this charging location directly belongs to. Admitted values: • HOTEL • RESTAURANT • CAFÉ • MALL • SUPERMARKET • SPORT • RECREATION_AREA • NATURE • MUSEUM • BIKE_SHARING • BUS_STOP • TAXI_STAND

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
			• TRAM_STOP • METRO_STATION • TRAIN_STATION • AIRPOORT • PARKING_LOT • CARPOOL_PARKING • FUEL_STATION • WIFI
opening_times	Structure	?	The times when the EVSEs at the location can be accessed for charging. The structure contains the following fields: • twentyfourseven (Boolean) • regular_hours: one or more structures with following fields: ○ weedkday (int - Monday =1,… Sunday = 7) ○ period_begin (in the format “HH:SS”) ○ period_end (in the format “HH:SS”) • exceptional_openings: one or more structures with following fields: ○ period_begin (in the format “HH:SS”) ○ period_end (in the format “HH:SS”) • exceptional_closings: structured as exceptional_openings
charging_when_closed	Boolean	?	Indicates if the EVSEs are still charging outside the opening hours of the location
images	Structure	*	Links to images related to the location such as photos or logos. The structure contains the following fields: • url • thumbnail • category (possible values: CHARGER, ENTRANCE, LOCATION) • type (string, max 4 characters) • width (int, max 5 characters) • height (int, max 5 characters)
energy_mix	Structure	?	Details on the energy supplied at this location. The structure contains the following fields: • is_green_energy (boolean) • energy_sources: structure with following fields: ○ source: enum value between NUCLEAR, GENERAL_FOSSIL, COAL, GAS, GENERAL_GREEN, SOLAR, WIND, WATER ○ percentage (value between 0 and 100) • environ_impact: structure with following fields: ○ category: enum value between NUCLEAR_WASTE e CARBON_DIOXIDE ○ amount (in g/kWh) • supplier_name (maximum length = 64 characters) • energy_product_name (maximum length = 64 characters)
last_updated	DateTime	1	Timestamp when this Location or one of its EVSEs or Connectors were last updated

Table 11: Locations module – response EVSE List parameters.

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
uid	String	1	Uniquely identifies the EVSE within the CPOs platform (maximum length of the string = 36 characters)
evse_id	String	*	Additional code, compliant with eMI3 standard version V1.0"Part 2: business objects."
status	Enum	1	Indicates the current status of the EVSE. Following values are foreseen: • AVAILABLE • BLOCKED • CHARGING • INOPERATIVE • OUTOFORDER • PLANNED • REMOVED • RESERVED • UNKOWN
status_schedule	Structure	*	Indicates a planned status update of the EVSE. The structure contains the following fields: • period_begin • period_end • status (see above enum values)
capabilities	Enum	*	List of functionalities that the EVSE is capable of. Following values are foreseen: • CHARGING_PROFILE_CAPABLE • CHARGING_PREFERENCES_CAPABLE • CHIP_CARD_SUPPORT • CONTACTLESS_CARD_SUPPORT • CREDIT_CARD_PAYABLE • DEBIT_CARD_PAYABLE • PED_TERMINAL • REMOTE_START_STOP_CAPABLE • RESERVABLE • RFID_READER • START_SESSION_CONNECTOR_REQUIRED • TOKEN_GROUP_CAPABLE • UNLOCK_CAPABLE
connectors	Structure	+	List of available connectors on the EVSE. See Table 12.
floor_level	String	?	Level on which the Charge Point is located (in garage buildings) in the locally displayed numbering scheme (maximum length of the string = 4 characters)
coordinates	Structure	?	Coordinates of the EVSE, expressed in the WGS 84 format in decimal degree. The structure contains the following fields: • Latitude • Longitude
physical_reference	String	?	A number/string printed on the outside of the EVSE for visual Identification (maximum length of the string = 16 characters)
directions	Structure	*	Multi-language human-readable directions when more detailed information on how to reach the EVSE from the Location is required. The structure contains the following fields: • language (language Code ISO 639-1) • text (maximum length = 512 characters)

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
parking_restrictions	Enum	*	The restrictions that apply to the parking spot. Admitted values: • EV_ONLY • PLUGGED • DISABLED • CUSTOMERS MOTORCYCLES
images	Structure	*	Links to images related to the EVSE such as photos or logos. The structure contains the following fields: • url • thumbnail • category (possible values: CHARGER, ENTRANCE, LOCATION) • type (string, max 4 characters) • width (int, max 5 characters) • height (int, max 5 characters)
last_updated	DateTime	1	Timestamp when this EVSE or one of its Connectors was last updated (or created).

Table 12: Locations module – response Connectors List parameters.

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
id	String	1	Identifier of the Connector within the EVSE (maximum length of the string = 36 characters)
standard	Enum	1	The standard of the installed connector. Following values are foreseen: • CHADEMO • CHAOJI • DOMESTIC_A • DOMESTIC_B • DOMESTIC_C • DOMESTIC_D • DOMESTIC_E • DOMESTIC_F • DOMESTIC_G • DOMESTIC_H • DOMESTIC_I • DOMESTIC_J • DOMESTIC_K • DOMESTIC_L • DOMESTIC_M • DOMESTIC_N • DOMESTIC_O • GBT_AC • GBT_DC • IEC_60309_2_single_16 • IEC_60309_2_three_16 • IEC_60309_2_three_32 • IEC_60309_2_three_64 • IEC_62196_T1 • IEC_62196_T1_COMBO • IEC_62196_T2 • IEC_62196_T2_COMBO

PARAMETER	TYPE	CARDINALITY	DESCRIPTION
			• IEC_62196_T3A • IEC_62196_T3C • NEMA_5_20 • NEMA_6_30 • NEMA_6_50 • NEMA_10_30 • NEMA_10_50 • NEMA_14_30 • NEMA_14_50 • PANTOGRAPH_BOTTOM_UP • PANTOGRAPH_TOP_DOWN • TESLA_R • TESLA_S
format	Enum	1	The format (socket/cable) of the installed connector. Following values are foreseen: • SOCKET • CABLE
power_type	Enum	1	Associated power table. Following values are foreseen: • AC_1_PHASE • AC_2_PHASE • AC_2_PHASE_SPLIT • AC_3_PHASE • DC
max_voltage	int	1	Maximum voltage of the connector (line to neutral for AC_3_PHASE), in volt [V].
max_amperage	int	1	Maximum amperage of the connector, in ampere [A].
max_electric_power	int	?	Maximum electric power that can be delivered by this connector, in Watts (W).
tariff_ids	Structure	*	List of Identifiers of the currently valid charging tariffs (maximum length of each string = 36 characters)
terms_and_conditions	URL	?	URL to the operator’s terms and conditions.
last_updated	DateTime	1	Timestamp when this Connector was last updated (or created).

Request object

With this kind of request it is possible to retrieve the information associated to a location, EVSE or connector. The structure of the API request is:

```
{locations_endpoint_url}/{location_id}[/{evse_uid}][/{connector_id}]
```

The response in this case is simply the requested object, with the above specified structure.