

---

# OIOI Documentation

*Release 4.16.3*

**PlugSurfing**

Sep 25, 2017



<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Business logic . . . . .	3
1.2	Fundamentals of OIOI . . . . .	3
<b>2</b>	<b>Sending a Message via OIOI</b>	<b>5</b>
2.1	Response . . . . .	6
<b>3</b>	<b>Test Accounts</b>	<b>9</b>
3.1	Users . . . . .	9
3.2	UIDs to test RFID . . . . .	9
<b>4</b>	<b>Glossary</b>	<b>11</b>
<b>5</b>	<b>Introduction for CPOs</b>	<b>15</b>
<b>6</b>	<b>Sending POI Data</b>	<b>17</b>
6.1	Sending Static Data . . . . .	17
6.2	Sending Dynamic Data . . . . .	18
<b>7</b>	<b>Sending CDRs</b>	<b>19</b>
<b>8</b>	<b>Receiving Remote Starts</b>	<b>21</b>
8.1	Starting a Session . . . . .	21
8.2	Stopping a Session . . . . .	21
<b>9</b>	<b>Allowing RFID Starts as CPO</b>	<b>23</b>
9.1	Online Authorization . . . . .	23
9.2	Offline Authorization . . . . .	23
<b>10</b>	<b>Introduction for OEMs</b>	<b>25</b>
10.1	Additional HTTP Header . . . . .	25
<b>11</b>	<b>Managing Users</b>	<b>27</b>
11.1	Register Users . . . . .	27
11.2	Verify Users (Log-In) . . . . .	27
11.3	User Details . . . . .	27
11.4	RFID Management . . . . .	28

<b>12</b>	<b>Receiving POIs</b>	<b>29</b>
12.1	Displaying stations in an area . . . . .	29
12.2	Displaying details of a station . . . . .	29
<b>13</b>	<b>Receiving CDRs</b>	<b>31</b>
<b>14</b>	<b>Sending Remote Starts</b>	<b>33</b>
14.1	Starting a Session . . . . .	33
14.2	Stopping a Session . . . . .	33
<b>15</b>	<b>Allowing RFID Starts as OEM</b>	<b>35</b>
15.1	Online Authorization . . . . .	35
15.2	Offline Authorization . . . . .	35
<b>16</b>	<b>Introduction for Fleet Operators</b>	<b>37</b>
<b>17</b>	<b>Receiving CDRs</b>	<b>39</b>
<b>18</b>	<b>Company Get Images</b>	<b>41</b>
18.1	Request . . . . .	41
18.2	Response . . . . .	41
18.3	Examples . . . . .	42
<b>19</b>	<b>Connector Post Status</b>	<b>43</b>
19.1	Request . . . . .	43
19.2	Response . . . . .	43
19.3	Examples . . . . .	44
<b>20</b>	<b>RFID Post</b>	<b>45</b>
20.1	Request . . . . .	45
20.2	Response . . . . .	46
20.3	Examples . . . . .	46
<b>21</b>	<b>RFID Verify</b>	<b>47</b>
21.1	Request . . . . .	47
21.2	Response . . . . .	48
21.3	Examples . . . . .	48
<b>22</b>	<b>Session Post</b>	<b>49</b>
22.1	Request . . . . .	49
22.2	Response . . . . .	50
22.3	Examples . . . . .	51
<b>23</b>	<b>Session Start</b>	<b>53</b>
23.1	Request . . . . .	53
23.2	Response . . . . .	54
23.3	Examples . . . . .	54
<b>24</b>	<b>Session Stop</b>	<b>57</b>
24.1	Request . . . . .	57
24.2	Response . . . . .	58
24.3	Examples . . . . .	58
<b>25</b>	<b>Station Get By IDs</b>	<b>59</b>
25.1	Request . . . . .	59
25.2	Response . . . . .	59

25.3	Examples . . . . .	63
<b>26</b>	<b>Station Get Surface</b>	<b>73</b>
26.1	Request . . . . .	73
26.2	Response . . . . .	74
26.3	Examples . . . . .	75
<b>27</b>	<b>Station Get Usage</b>	<b>77</b>
27.1	Request . . . . .	77
27.2	Response . . . . .	77
27.3	Examples . . . . .	78
<b>28</b>	<b>Station Post</b>	<b>79</b>
28.1	Request . . . . .	79
28.2	Response . . . . .	82
28.3	Examples . . . . .	82
<b>29</b>	<b>User Add Credit Card</b>	<b>85</b>
29.1	Request . . . . .	85
29.2	Response . . . . .	85
29.3	Examples . . . . .	86
<b>30</b>	<b>User Add Payment</b>	<b>87</b>
30.1	Request . . . . .	87
30.2	Response . . . . .	88
30.3	Examples . . . . .	88
<b>31</b>	<b>User Add RFID</b>	<b>89</b>
31.1	Request . . . . .	89
31.2	Response . . . . .	90
31.3	Examples . . . . .	90
<b>32</b>	<b>User Block RFID</b>	<b>91</b>
32.1	Request . . . . .	91
32.2	Response . . . . .	92
32.3	Examples . . . . .	92
<b>33</b>	<b>User Change Password</b>	<b>93</b>
33.1	Request . . . . .	93
33.2	Response . . . . .	93
33.3	Examples . . . . .	94
<b>34</b>	<b>User Charging Key Activate</b>	<b>95</b>
34.1	Request . . . . .	95
34.2	Response . . . . .	95
34.3	Examples . . . . .	96
<b>35</b>	<b>User Get Bills</b>	<b>97</b>
35.1	Request . . . . .	97
35.2	Response . . . . .	98
35.3	Examples . . . . .	98
<b>36</b>	<b>User Get Charging Keys</b>	<b>101</b>
36.1	Request . . . . .	101
36.2	Response . . . . .	101
36.3	Examples . . . . .	102

<b>37 User Get Details</b>	<b>103</b>
37.1 Request . . . . .	103
37.2 Response . . . . .	103
37.3 Examples . . . . .	104
<b>38 User Get Recent Sessions</b>	<b>107</b>
38.1 Request . . . . .	107
38.2 Response . . . . .	107
38.3 Examples . . . . .	108
<b>39 User Get Recent Stations</b>	<b>111</b>
39.1 Request . . . . .	111
39.2 Response . . . . .	111
39.3 Examples . . . . .	112
<b>40 User Logout</b>	<b>115</b>
40.1 Request . . . . .	115
40.2 Response . . . . .	115
40.3 Examples . . . . .	116
<b>41 User Post Details</b>	<b>117</b>
41.1 Request . . . . .	117
41.2 Response . . . . .	118
41.3 Examples . . . . .	119
<b>42 User Register</b>	<b>121</b>
42.1 Request . . . . .	121
42.2 Response . . . . .	121
42.3 Examples . . . . .	122
<b>43 User Reset</b>	<b>123</b>
43.1 Request . . . . .	123
43.2 Response . . . . .	123
43.3 Examples . . . . .	123
<b>44 User Unblock RFID</b>	<b>125</b>
44.1 Request . . . . .	125
44.2 Response . . . . .	126
44.3 Examples . . . . .	126
<b>45 User Verify</b>	<b>127</b>
45.1 Request . . . . .	127
45.2 Response . . . . .	127
45.3 Examples . . . . .	128
<b>46 Get All Vehicles</b>	<b>129</b>
46.1 Request . . . . .	129
46.2 Response . . . . .	129
46.3 Examples . . . . .	130



**Warning:** This is the latest version of the OIOI, **version 4**.

For other versions, e.g. **version 3**, open the menu on the very bottom left of the screen.

Even though this version is marked as *stable*, new fields may be added to requests and responses.

Your service must still accept all responses. You may ignore the additional fields or implement an update.

Requests will only get additional fields if they are optional.

The main documentation for the OIOI is organized into a couple of sections:

- *Introduction*
- *Connecting as CPO*
- *Connecting as OEM*
- *Connecting as Fleet Operator*
- *HTTP Calls*





To enhance the speed of onboarding *CPOs* and OEMs, PlugSurfing has created an open API: the OIOI. The OIOI covers all messages between a CPO and an EMP or a *Roaming Platform* and an OEM.

### Business logic

The aim of the OIOI is to allow electric car drivers access to any charging station around. The OIOI therefore has two functions. First, to on-board CPOs into an EMP's network. Second, to allow communication with the charging stations from the electric car driver's point of view.

Hence, CPOs can get access to many electric car drivers and electric car drivers can get access to many charging station networks. For both sides of the spectrum the OIOI becomes an easy gateway to more customers and more comfort.

The idea of the OIOI came up after many, many CPOs wanted to connect to Plugsurfing, but did not have the technical means (no API) to connect. The OIOI helps them now to connect to Plugsurfing. On the other hand, it saves money and time of all parties that are involved; the workload of connecting many different interface is, of course, significant.

The main difference between the OIOI and other existing API definitions is:

- it is open to anyone to use or distribute it free of charge
- it is very simple to implement

### Fundamentals of OIOI

The OIOI consists of several main functions that establish a relationship between the electric car driver, the EMP, and the charging station network.

The following integration is required:

**Point of Interest** *POI* includes the static and dynamic information regarding the whereabouts and properties of a charging station. This is necessary for an *EV* Driver to find a charging station using a smartphone app.

**Charge Detail Records** A *CDR* informs about who has been charging at which station, as well as the consumption. A CDR provides objective, raw data allowing the payment and clearing of funds within the value chain.

**Client Exchange** In order for a CPO to allow or deny access of a customer using RFID, a unique identifier between the EMP and the CPO needs to be exchanged. The identifiers are made out of a contract ID (*EVCO ID*) and the unique UID of the charging key.

**Remote Start/Stop** In order to start or stop a charging session from an app, both partners implement the remote start/stop functionality of the OIOI. The EMP implements it client side, while the CPO implements it server side.

**API Key** In order to have access to an OIOI service API, you need to be in possession of an API key. You supply that key with every request and the receiver verifies that you have access to the requested resource.

---

**Note:** If you want to get an API key to connect to PlugSurfing, please contact the [PlugSurfing Service](#).

---

<p><b>Warning:</b> Never give your API key to anyone except as part of an OIOI request! No one will ever ask you for your API key! PlugSurfing may cancel your access if your key is associated with malicious behaviour.</p>
---

---

### Sending a Message via OIOI

---

To send a message, your server issues a POST request to:

```
https://api.plugsurfing.com/api/v4/request
```

---

**Note:** The staging server for testing is located at <https://dev-api.plugsurfing.com/api/v4/request>

---

**Important:** All the concepts explained here apply also to any server instance that you may create. If you expect any party to send requests to you using OIOI, make sure your servers accept calls as described here.

---

A message request is made of 2 parts: HTTP header and HTTP body. The HTTP header must contain the following headers:

```
Authorization: key=<your-API-key>
Content-Type: application/json
```

The HTTP body must contain the JSON request. Example HTTP request:

```
Content-Type: application/json
Authorization: key=a39ff3fb-fe0a-40a3-bdde-df6372c07c89

{
  "station-get-by-center-radius": {
    "center-lat": 52.5167,
    "center-long": 13.3833,
    "radius": 100,
    "unit": "km"
  }
}
```

---

**Important:** OIOI is not a REST interface. Instead, it is simple JSON over HTTP. All requests are sent to the same

---

endpoint. The receiver will know which resource to use based on the request body.

---

**Important:** All data must be UTF-8 encoded.

---

## Response

If the request was valid and successful, the response will contain the requested data or information about the success. Otherwise, the HTTP response code will not be 200 and more information about the error will be inside the response body.

Possible HTTP response codes and their meanings:

Response	Description
200	<b>Success.</b> The request was handled successfully. The response body will contain more details about the message status in JSON format.
4xx	<b>Error.</b> Something was wrong with your request. The response body will contain more information in JSON format. The property error will contain more detailed information as to what was wrong.
400	<b>Bad request.</b> Your request contains data and/or is in a format that is not allowed.
401	<b>Unauthorized.</b> Your request contains user data that is not authenticated or authorized.
403	<b>Forbidden.</b> You do not have access to this method and/or resource.
404	<b>Not Found.</b> One or more of the requested entities were not found.
5xx	<b>Server Error.</b> Errors in the 500-599 range (such as 500 or 503) indicate that there was an internal error in the server while trying to process the request, or that the server is temporarily unavailable (for example, because of timeouts). You must retry later, honoring any Retry-After header included in the response. Your services must implement exponential back-off.

## OIOI Response Codes

---

**Note:** OIOI Error codes are not part of the OIOI response, yet. These codes will be added to the OIOI shortly.

---

A more general code must always be accepted in parallel to the more specific code. For example: If a customer successfully starts a charging session, the code 000 is valid, as well as the more specific 011. The same is true for errors. If a client wants to authenticate a customer, any 14x error could be returned. At the same time, a 100 could also be returned and must be accepted by the client.

Error Code	Description
0xx	<b>Success</b>
000	Success
011	Successfully started a charging session The customer is charging at the EVSE
012	Successfully authorized a charging session The customer must now plug in the cable to start
1xx	<b>PlugSurfing Errors</b>
100	System error
101	Database error
102	System timeout
140	Authentication failed: No positive authentication response
141	Authentication failed: Invalid email or password
142	Authentication failed: Invalid email
143	Authentication failed: Email already exists
144	Authentication failed: Email does not exist
145	Authentication failed: User token not valid
180	Entity not found
181	EVSE not found
182	Session not found
183	Company not found
184	Vehicle not found
185	Subscription plan not found
186	Group not found
187	EVSE ID does not support direct pay
188	EVSE ID does not support remote stop
190	EVCO ID error
191	EVCO ID not found
192	EVCO ID locked
193	EVCO ID has no valid payment method
2xx	<b>Client Error</b>
200	Client request error
210	Invalid API key
211	Invalid partner identifier
220	API key not allowed to access the requested resource
230	Invalid request format
3xx	<b>Operator and EVSE Errors</b>
300	System error
302	System timeout
310	EVSE error
312	EVSE timeout
320	EVSE already in use
321	No EV connected to EVSE
4xx	<b>Hub Errors</b>
400	System error
402	System timeout
8xx	<b>Payment Provider Errors</b>
800	System error
802	System timeout
805	This user is not allowed to use this method

Continued on next page

Table 2.1 – continued from previous page

Error Code	Description
830	Invalid format
850	Invalid payment method
860	Bank transfer error
861	Bank account not valid
862	Invalid name
863	Invalid IBAN
864	Invalid BIC
870	Credit card error
871	Credit card not valid
872	Invalid card holder name
874	Invalid credit card number
875	Invalid expiration date
876	Invalid CVC
880	PayPal error

On the staging environment, you can use the following users and UIDs for testing.

### Users

The staging environment has two existing users that you can use for testing.

1. **test1@plugsurfing.com**
  - Password: `test`
  - Has a payment method
  - Has a charging key (UID 04523022AA4881)
2. **test2@plugsurfing.com**
  - Password: `test`
  - Has no payment method
  - Has no charging key

### UIDs to test RFID

The staging environment holds the following UIDs for testing.

1. **04523022AA4881**
  - Is active
  - Belongs to a user (`test1@plugsurfing.com`)
2. **0477BD22AA4880**
  - Is inactive

- Does not belong to a user

3. **0477BD22AA4AAA**

- Does not exist



**EV** Electric Vehicle.

[Wikipedia EV](#)

**Charging Station** A place where a driver of an *EV* can charge their car, can vary in size from very small to pretty large. Connectors can supply electricity either as alternating current (AC) or direct current (DC).

[Wikipedia Charging Station](#)

**POI** Point of Interest. POI refers to the whereabouts and attributes of a charging station, including latitude, longitude, plug type, and charging speed. Throughout this documentation, the terms *charging stations* and POI are used interchangeably.

**EVSE** Electric Vehicle Supply Equipment. A *charging station* may have one or more EVSE IDs. Usually, one EVSE ID uniquely identifies one connector.

There are cases where one EVSE ID identifies multiple connectors. In that case, all connectors with the same EVSE ID always share the same status (e.g. *Available* or *Occupied*). That also means that only one connector of all the connectors sharing the same EVSE ID can be in use at any one point in time.

Example: DE\*123\*E12345\*A

An EVSE ID must be valid with respect to the following structure:

- `[A-Za-z]{2}\*[A-Za-z0-9]{3}\*?E[A-Za-z0-9\*]{1,30}`
- Country Code (international two-letter code in accordance with [ISO 3166-1 alpha-2](#))
- The literal `*` (optional)
- CPO identifier (letters and numbers; three characters uniquely identifying the CPO of the EVSE)
- The literal `*` (optional)
- The literal `E`
- ID of the EVSE (letters, numbers, and the literal `*`; 1 - 30 characters)

---

**Note:** When you are based in Germany, please see here for more general information and details on how to register a CPO identifier: [BDEW Codes](#).

---

See also [Wikipedia Charging Station](#).

**CPO** Charge Point Operator. The CPO manages, operates, and maintains charging station infrastructure.

**EMP** E-Mobility Provider. Companies that provide *EV* drivers with access to charging stations. PlugSurfing, as interface between electric car drivers and CPOs, is an EMP.

**Roaming Platform** [PlugSurfing](#) is a roaming platform.

**EVCO ID** Contract ID of *EMP* and *EV* driver. Uniquely identifies the customer across all networks.

An EVCO ID must be valid with respect to the following structure:

- $[A-Za-z] \{2\} [\*|-]? [A-Za-z0-9] \{3\} [\*|-]? [A-Za-z0-9] \{6\} [\*|-]? [\d|X]$
- Country Code (international two-letter code in accordance with [ISO 3166-1 alpha-2](#))
- The literal \* **or** the literal - (optional)
- EMP identifier (letters and numbers; three characters uniquely identifying the EMP of the EVCO)
- The literal \* **or** the literal - (optional)
- ID of the EVCO (letters and numbers; six characters)
- The literal \* **or** the literal - (optional)
- Check digit (A number or the literal X)

---

**Note:** When you are based in Germany, please see here for more general information and details on how to register an EMP identifier: [BDEW Codes](#).

---

**CDR** Charge Detail Record. After a customer finished charging at a *charging station*, the *CPO* provides the EMP with the CDR for that session. Includes data like:

- Session start date/time
- Session end date/time
- Consumed energy
- EVCO ID or UID

---

**Note:** A CDR may be sent by a CPO before the session finished. For example to inform the EMP of a started session.

---

**RFID** An RFID token that authenticates an *EV* driver at a *charging station*. Common RFID carriers are cards (credit card format) and key hangers.

**Static data** Data on the charging station that doesn't change frequently. Charging station location, address, connector type, etc.

**Dynamic data** Data that may change frequently, like the status of a connector.

**Partner Identifier** A universally unique identifier that identifies the partner who issues an API call. This is different from an API key! The sending partner chooses the identifier and provides it to the receiving partner in a secure manner. Must be unique and hard to guess. Must be a random string that is at least 16 characters long.

A company with one API key can use multiple partner identifiers, for example to make API calls for another company.

At the same time, multiple API keys can use the same partner identifier to act on behalf of that entity.



---

## Introduction for CPOs

---

In order for a CPO to connect to an EMP and make their charging stations available in the EMP's network, the following steps need to be taken:

- *Sending POI Data*
- *Sending CDRs*
- *Receiving Remote Starts*
- *Allowing RFID Starts as CPO*

**Warning:** If sessions in your network are stoppable, please inform the EMP. The EMP needs to make sure that their services allow for stopping accordingly.



---

## Sending POI Data

---

In order to add your stations to the network, The EMP has to know about your charging station locations and attributes.

POI is split into two sections:

- *Sending Static Data*
- *Sending Dynamic Data*

Static data is data that does not change very often, including for example the address and latitude/longitude of a station.

Dynamic data is the name for the statuses of the connectors. This will show the EV driver whether a connector is currently available or not.

### Sending Static Data

The CPO must send all stations via OIOI. In the beginning, there will be a one-time push of all existing charging stations. From then on, only new stations and station deletions will be pushed.

It is possible that a connected CPO will send stations from multiple CPOs. It is also possible that the EMP already has some of those CPOs listed in their database. Therefore, the EMP needs to be able to associate the pushed stations' CPOs with existing CPOs in the database. Before the CPO pushes their first station, they need to inform the EMP about their internal identifiers for all given CPOs. Then, when the CPO pushes a station with a certain CPO identifier that may be unique to the CPO's system, the EMP knows which CPO to pick from their database.

**Summary:** For a new CPO, pushing POI data is done in three steps:

1. Setting up CPOs
2. Full load of all existing charging stations (push)
3. Continuously sending updates for new and existing charging stations (push)
  - **Role:** Sender
  - **Implementation:** *Station Post*

## Sending Dynamic Data

When the status of a connector changes, the CPO must send a `connector-post-status` request to push the change.

---

**Note:** PlugSurfing only accepts dynamic data changes if they are associated to stations that were pushed by the same partner.

---

- **Role:** Sender
- **Implementation:** *Connector Post Status*



## CHAPTER 7

---

### Sending CDRs

---

CDRs are the basis of all billing and clearing. A CDR contains all required information about a session, e.g. the customer or the consumed energy.

---

**Note:** You can also use this method to send information about sessions that just started or are still ongoing. To do this, simply omit optional data that is not available at the moment, e.g. the session interval's stop time.

---

- **Role:** Sender
- **Implementation:** *Session Post*



---

## Receiving Remote Starts

---

### Starting a Session

Whenever an EV driver wants to start a session in your network, the EMP will send a `session-start` request. The EMP will analyze the CPO's response and must display an appropriate message to the user.

- **Role:** Receiver
- **Implementation:** *Session Start*

### Stopping a Session

Whenever an EV driver wants to stop a running session, the EMP will send a `session-stop` request.

- **Role:** Receiver
- **Implementation:** *Session Stop*



---

## Allowing RFID Starts as CPO

---

There are two options to allow RFID charging at a CPO:

- *Online Authorization*
- *Offline Authorization*

With online authorization, a UID is verified every time a customer wants to charge at one of the CPO's stations.

With offline authorization, the EMP must send a whitelist of known UIDs to the CPO.

### Online Authorization

With online authorization, the CPO has to send a verification request to the EMP every time an RFID token is used at one of the CPO's stations. Depending on the EMP's response, the EMP does or does not allow charging for the given UID. If an EMP authorizes the UID, the CPO can allow the session and subsequently send the correlating CDR to that EMP. See also *Sending CDRs*.

- **Role:** Sender
- **Implementation:** *RFID Verify*

### Offline Authorization

With offline authorization, the EMP will send all known UIDs to the CPO in regular intervals, e.g. once per day. It is then the CPO's responsibility to authorize any UID and send the subsequent CDR to the corresponding EMP, if the UID was part of the last push of UIDs from the EMP to the CPO. See also *Sending CDRs*.

- **Role:** Receiver
- **Implementation:** *RFID Post*



---

## Introduction for OEMs

---

In order for an OEM to connect to a *Roaming Platform* and make their charging stations available in the OEM's network, the following steps need to be taken:

- *Managing Users*
- *Receiving POIs*
- *Receiving CDRs*
- *Sending Remote Starts*
- *Allowing RFID Starts as OEM*

### Additional HTTP Header

For basic info, see *Sending a Message via OIOI*.

Example HTTP request:

```
Content-Type: application/json
Authorization: key=a39ff3fb-fe0a-40a3-bdde-df6372c07c89

{
  "user-verify": {
    "username": "your-user",
    "password": "mypassword",
    "is-token": false,
    "language": "en"
  }
}
```





As an OEM you can manage your users on the platform. For example registration and log-in.

After user registration or log-in, you will get a token in the response. That token will be used in future calls to authenticate the user, so that your application does not store any user passwords.

### Register Users

To register a new user on the platform, send this call.

- **Role:** Sender
- **Implementation:** *User Register*

### Verify Users (Log-In)

If you want to verify an existing user by asking for the password, you can use this call.

- **Role:** Sender
- **Implementation:** *User Verify*

### User Details

If you want to manage additional user data, like for example the address, you can send that data to the platform.

- **Role:** Sender
- **Implementation:** *User Post Details*

Later it is possible to get the data, e.g. for displaying in an app.

- **Role:** Sender

- **Implementation:** *User Get Details*

## RFID Management

To register a new RFID medium for a user, do the following.

- **Role:** Sender
- **Implementation:** *User Add RFID*

It is possible that a user lost the RFID medium and wants to block it. This call allows a user to do so.

- **Role:** Sender
- **Implementation:** *User Block RFID*

---

**Note:** It may take up to 24h until the block has propagated throughout all networks.

---

---

## Receiving POIs

---

As an OEM, you can connect to a *Roaming Platform* like PlugSurfing to display their POI data and allow your customers to charge at the Roaming Platform's charging stations. For example to display charging stations on an app's map.

---

**Note:** For OEMs, OIOI does not offer a full download of all available data. Please use the provided methods to display geographically limited data to your users.

---

---

**Note:** The total number of stations returned per request is limited, regardless of the geographical dimensions.

---

**Warning:** A *Roaming Platform* may cancel your access if your API key is associated with malicious behaviour.

## Displaying stations in an area

To display basic data about stations on a map or in a list, you can get stations by providing a surface (via latitude/longitude).

- **Role:** Sender
- **Implementation:** *Station Get Surface*

## Displaying details of a station

If you want to display more detailed information, you can query for those by a station's ID.

- **Role:** Sender

- **Implementation:** *Station Get By IDs*

# CHAPTER 13

---

## Receiving CDRs

---

CDRs are the basis of all billing and clearing. A CDR contains all required information about a session, e.g. the customer or the consumed energy.

---

**Note:** A *Roaming Platform* can also use this method to send information about sessions that just started or are still ongoing. To do this, the Roaming Platform omits optional data that is not available at the moment, e.g. the session interval's stop time.

---

- **Role:** Receiver
- **Implementation:** *Session Post*



### Starting a Session

Whenever an EV driver wants to start a session in the *Roaming Platform's* network, the OEM will send a `session-start` request. The OEM must analyze the Roaming Platform's response and must display an appropriate message to the user.

- **Role:** Sender
- **Implementation:** *Session Start*

### Stopping a Session

Whenever an EV driver wants to stop a running session, the OEM must send a `session-stop` request.

Please note that in order for a session to be stoppable, the response to `session-start` must include `"is-stoppable": true`. See also *Session Start*.

- **Role:** Sender
- **Implementation:** *Session Stop*





---

## Allowing RFID Starts as OEM

---

There are two options to allow RFID charging as an OEM:

- *Online Authorization*
- *Offline Authorization*

With online authorization, a UID is verified every time a customer wants to charge.

With offline authorization, the OEM must send a whitelist of known UIDs to the *Roaming Platform*.

### Online Authorization

With online authorization, the *Roaming Platform* will send a verification request to the OEM every time an RFID token is used at one of the stations in the CPO's network. Depending on the OEM's response, the Roaming Platform does or does not allow charging for the given UID. If an OEM authorizes the UID, the Roaming Platform can allow the session and subsequently send the correlating CDR to that OEM. See also *Receiving CDRs*.

- **Role:** Receiver
- **Implementation:** *RFID Verify*

### Offline Authorization

With offline authorization, the OEM will send all known UIDs to the *Roaming Platform* in regular intervals, e.g. once per day. It is then the Roaming Platform's responsibility to authorize any UID and send the subsequent CDR to the correct OEM, if the UID was part of the last push of UIDs from that OEM to the Roaming Platform. See also *Receiving CDRs*.

- **Role:** Sender
- **Implementation:** *RFID Post*



---

## Introduction for Fleet Operators

---

You should first read the *Introduction*. Please see the *Glossary* if any terms are unclear.

As a fleet operator or leasing company you are interested in maximum transparency of the charging behaviour of your clients. If your clients charge through an *EMP* that has implemented the OIOI, the EMP can forward any charging data to you directly. This means you will always have the newest data available to analyze and present to your customers.

Please note that the fleet operators or leasing companies will always be on the receiving side. Other parties like EMPs will push *CDRs* to the fleet operator or leasing company in regular intervals or when they receive it.

In order for a fleet operator or leasing company to receive CDRs automatically, the following calls must be implemented by that company:

- *Receiving CDRs*



# CHAPTER 17

---

## Receiving CDRs

---

CDRs are the basis of all billing and clearing. A CDR contains all required information about a session, e.g. the customer or the consumed energy.

---

**Note:** This method can be used to send information about sessions that just started or are still ongoing. To do this, optional data that is not available at the moment, e.g. the session interval's stop time, will be omitted.

---

- **Role:** Receiver
- **Implementation:** *Session Post*



---

## Company Get Images

---

### Request

"company-get-images" identifies the call as a company-get-images call.

### Fields

**company-id** This field identifies the company (integer).

### Response

#### Fields

**company** This field contains the name of the company (string).

**urls** This field contains all URLs that point to images of the company (array of strings).

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**183** Company not found

## Examples

Request:

```
{
  "company-get-images": {
    "company-id": 123
  }
}
```

Response:

```
{
  "company": "ACME Inc.",
  "urls": [
    "https://s3.amazonaws.com/ksr/projects/392582/photo-main.jpg",
    "https://commons.wikimedia.org/wiki/File:Acme_anvil.gif#/media/File:Acme_
↔anvil.gif"
  ],
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



---

## Connector Post Status

---

### Request

"connector-post-status" identifies the call as a status update call.

### Fields

**connector-id** The EVSE ID of the connectors (string). All connectors with the given connector-id will be updated. See also *EVSE*.

**status** One of the following strings:

- "Available"
- "Occupied"
- "Offline"
- "Reserved"
- "Unknown"

**partner-identifier** The partner identifier of the partner that owns the connector (string). See also *partner identifier*

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**181** EVSE not found

## Examples

Request:

```
{
  "connector-post-status": {
    "connector-id": "DE*8PS*E123456",
    "partner-identifier": "123456-123456-abcdef-abc123-456def",
    "status": "Available"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

# CHAPTER 20

---

## RFID Post

---

---

**Note:** UIDs are read from left to right using big-endian format.

---

---

**Note:** Valid lengths of UIDs are 8, 14, and 20 characters. If you have shorter UIDs, you need to zero-pad them on the left.

---

---

**Note:** UIDs need to be all upper case.

---

RFID Post acts like a complete whitelist of UIDs on every request. Every call has to include the complete list of RFIDs that shall be active.

All RFIDs that were pushed before with the same "partner-identifier" and are not part of a request must be made invalid on the receiver side.

## Request

"rfid-post" identifies this call as an rfid-post call.

## Fields

**rfids** An array of strings. Each string represents one RFID (UID).

---

**Important:**

- An RFID must have a length of 8, 14 or 20 characters. If necessary, the RFID must be zero-padded on the left.

- It should be read from left to right using big-endian format.
  - All characters must be upper-case.
- 

**partner-identifier** The partner identifier of the partner that shall be associated with these UIDs (string). See also *partner identifier*

## Response

### Fields

**processed** The number of new UIDs added by this call (integer).

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

## Examples

Request:

```
{
  "rfid-post": {
    "rfids": [
      "12345678",
      "abcdefab"
    ],
    "partner-identifier": "123456-123456-abcdef-abc123-456def"
  }
}
```

Response:

```
{
  "rfid": {
    "processed": 2
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

---

## RFID Verify

---

---

**Note:** UIDs are read from left to right using big-endian format.

---

---

**Note:** Valid lengths of UIDs are 8, 14, and 20 characters. If you have shorter UIDs, you need to zero-pad them on the left.

---

---

**Note:** UIDs need to be all upper case.

---

### Request

"rfid-verify" identifies this call as an rfid verify call.

### Fields

**rfid** The RFID (UID) to verify (string).

---

**Important:**

- An RFID must have a length of 8, 14 or 20 characters. If necessary, the RFID must be zero-padded on the left.
  - It should be read from left to right using big-endian format.
  - All characters must be upper-case.
-

## Response

### HTTP Status code

**200 OK** The request was processed successfully.

### Result code

**0** Success

**191** EVCO ID not found

**192** EVCO ID locked

**193** EVCO ID has no valid payment method

## Examples

Request:

```
{
  "rfid-verify": {
    "rfid": "12345678ABCDEF"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"session-post" identifies the call as a session-post call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**session-id** A unique ID that identifies this session (string).

The session ID must be a globally unique identifier.

---

**Note:** If you are the CPO and owner of the session, this identifier must be created by you.

---

**connector-id** The EVSE ID of the connector where the session took place (string).

**session-interval** The start and, optionally, stop time of the session (object).

For example the time the driver plugged in and out of the station.

**start** The time the session started (string).

The date/time format is RFC3339 (Y-m-d\TH:i:sP).

**stop** The time the session stopped (string).

The date/time format is RFC3339 (Y-m-d\TH:i:sP).

**charging-interval (optional)** The start and stop time of charging (object).

For example the time the charging first started and last ended.

**start** The time the session started (string).

The date/time format is RFC3339 (Y-m-d\TH:i:sP).

**stop** The time the session stopped (string).

The date/time format is RFC3339 (Y-m-d\TH:i:sP).

**energy-consumed (optional)** The consumed energy in kWh (float).

**calculated-cost (optional)** The cost of the session (object).

**amount** The cost amount (float).

**currency** The currency of the cost (string).

E.g. "EUR".

**partner-identifier** The partner identifier of the partner that shall be associated with this CDR. See also *partner identifier*

## Response

### Fields

**reason** If success was false, reason explains what the problem was. This field is of type string. Will be null on success.

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**100** System error



## Examples

Request:

```
{
  "session-post": {
    "user": {
      "identifier": "12345678",
      "identifier-type": "rfid"
    },
    "session-id": "abcdef-123456-abc123-456def",
    "connector-id": "DE*8PS*ETABCD*1",
    "session-interval": {
      "start": "2010-01-01T11:00:00+00:00",
      "stop": "2010-01-01T17:00:00+00:00"
    },
    "charging-interval": {
      "start": "2010-01-01T12:00:00+00:00",
      "stop": "2010-01-01T16:00:00+00:00"
    },
    "energy-consumed": 16.5,
    "calculated-cost": {
      "amount": 14.32,
      "currency": "EUR"
    },
    "partner-identifier": "123456-123456-abcdef-abc123-456def"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



### Request

"`session-start`" identifies the call as a session-start call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**Warning:** For a session start request, it is usually required that the "`identifier-type`" **must be** "`evco-id`".

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**connector-id** The EVSE ID that identifies the connector where the session should take place (string).

**payment-reference (optional)** Identifies the chosen payment reference the user wants to use to pay for this session (string).

## Response

### Fields

**is-stoppable (optional)** Indicates whether the session can be stopped via “session-stop” API call (boolean).

**session-id (optional)** The session id of the started session (string).

**Warning:** Depending on the CPO’s requirements, a `session-id` may be mandatory if the session is stoppable.

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

- 0** Success
- 140** Authentication failed: No positive authentication response
- 144** Authentication failed: Email does not exist
- 145** Authentication failed: User token not valid
- 181** EVSE not found
- 300** CPO error
- 302** CPO timeout
- 310** EVSE error
- 312** EVSE timeout
- 320** EVSE already in use
- 321** No EV connected to EVSE

## Examples

Request:

```
{
  "session-start": {
    "user": {
      "identifier-type": "evco-id",
      "identifier": "DE*8PS*123456*7"
    },
    "connector-id": "1356",
    "payment-reference": "1212"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

Stoppable response:

```
{
  "session-start": {
    "session-id": "abcdef-123456-abc123-456def",
    "is-stoppable": true
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



### Request

"`session-stop`" identifies the call as a session-stop call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**Warning:** For a session stop request, it is usually required that the "`identifier-type`" **must be** "`evco-id`".

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**connector-id** The EVSE ID that identifies the connector where the session should be stopped (string).

**session-id (optional)** A unique ID that identifies this session (string).

**Warning:** Depending on the CPO's requirements, a `session-id` may be mandatory.

## Response

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

**181** EVSE not found

## Examples

Request:

```
{
  "session-stop": {
    "user": {
      "identifier-type": "evco-id",
      "identifier": "DE*8PS*123456*7"
    },
    "connector-id": "1356",
    "session-id": "dfdf"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



---

## Station Get By IDs

---

### Request

"station-get-by-ids" identifies the call as a station-get-by-ids call.

### Fields

**station-ids** An array of IDs (integers).

The maximum number of IDs per call is ten.

### Response

#### Fields

Optional fields may be omitted or have the value `null`.

**stations** An array of charging stations (objects).

**id** The responder's internal ID of the station (integer)

**address (optional)**

**street (optional)** string

**street-number (optional)** string

**city (optional)** string

**zip (optional)** string

**country (optional)** string

**contact (optional)**

**email (optional)** string

**web (optional)** string

**phone (optional)** string

**fax (optional)** string

**operator-company-id (optional)** The CPO of the charging station.

The charge point operator is the company responsible for the functioning of the station. Access to the station usually also goes through the CPO.

**operator-logo (optional)** URL to the logo image (string).

**floor-level (optional)** On which floor the station is located, for example in a parking house (integer).

**is-free-charge** Whether charging can be done without cost (boolean).

**last-static-change (optional)** string (format: "2016-05-09T04:08:06+02:00")

**last-dynamic-change (optional)** string (format: "2016-05-09T04:08:06+02:00")

**name** string

**description (optional)** string

**latitude** float

**longitude** float

**open-hour-notes (optional)** An array of objects containing certain opening periods:

**times** Opening and closing time (array of strings).

**days** Weekdays when the interval starts and ends (array of two strings).

Both are the same if it is for one specific day only.

Example:

```
{
  "open-hour-notes": [
    {
      "times": [
        "07:30",
        "19:00"
      ],
      "days": [
        "Mo",
        "Fr"
      ]
    },
    {
      "times": [
        "09:00",
        "15:00"
      ],
      "days": [
        "Sa",
        "Sa"
      ]
    }
  ]
}
```

This example means the following: For the interval Monday to Friday, the station is open from 07:30 to 19:00. On Saturday, the station is open from 09:00 to 15:00.

**total-parking** The number of parking spots that are available at the station (integer).

**notes (optional)** Additional notes, for example how to find the station (string).

**is-green-power-available** boolean

**is-plugin-charge** boolean

**is-roofed** boolean

**is-reservable** boolean

**has-dynamic-info** boolean

**is-open-24** boolean

**dynamic-status-summary (optional)** Whether the station is currently available (string).

One of:

- "Available"
- "Occupied"
- "Reserved"
- "Offline"
- "Unknown"

**is-validated** boolean

**is-private** Whether the station is privately owned (boolean).

For details, please contact the connected partner.

**owner-company-id (optional)** The owner of the charging station.

The owner is usually either the CPO or something like a restaurant or Ikea, owning the stations on their property.

**service-providers (optional)** An array of all service providers of the charging station.

A service provider is a company that grants access to a charging station. See *EMP*.

**connectors (optional)**

**id** The responder's internal ID of the station (integer)

**status (optional)** Whether the connector is currently available (string).

One of:

- "Available"
- "Occupied"
- "Reserved"
- "Offline"
- "Unknown"

**last-change (optional)** string (format: "2016-05-09T04:08:06+02:00")

**name (optional)** The type of connector (string).

One of:

- "UNKNOWN"
- "Type1"
- "Type2"
- "Type3"
- "Schuko"
- "Combo"
- "CeeBlue"
- "CeeRed"
- "Cee2Poles"
- "CeePlus"
- "3PinSquare"
- "Chademo"
- "Tesla"
- "Scame"
- "Nema5"
- "T13"
- "T15"
- "T23"
- "Marechal"
- "TypeE"

**speed (optional)** Max. available charging speed of the connector (string).

**mode (optional)** string

**external-id (optional)** If available, the EVSE ID of the connector (string). See also *EVSE*.

If an EVSE ID is not available, another ID provided by the CPO may be returned.

**prices (optional)** Prices for charging at this connector (object). The prices of a connector always override the prices of a station.

Connector prices may also be returned if they equal the station prices.

**starting-fee** The fee of starting a session at this connector (string; format "1.23").

**charging-per-hour** The fee of charging energy at this connector, per hour (string; format "1.23").

**parking-per-hour** The fee of parking with a connection to this connector, per hour (string; format "1.23").

**charging-per-kwh** The fee of charging energy at this connector, per kWh (string; format "1.23").

**currency** The currency of the prices (string; format "EUR").

**companies** An array of companies.

This array lists all companies that are relevant for the returned stations.

**id** The internal id of the company (integer).

The id of the company relates to the following fields in stations:

- operator-company-id
- owner-company-id
- service-providers

**name** The name of this company, e.g. “PlugSurfing”.

**contact** Available methods of contact.

**email** string or null.

**web** string or null.

**phone** string or null.

**fax** string or null.

**address**

**street** string or null.

**street-number** string or null.

**city** string or null.

**zip** string or null.

**country** string or null.

**description** A description (string or null).

**type** The type of the company (string or null).

E.g. “hotel”.

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

## Examples

Request:

```
{
  "station-get-by-ids": {
    "station-ids": [
      1770,
      1169,
      1003,
      2057
    ]
  }
}
```

Response:

```
{
  "stations": [
    {
      "id": 1003,
      "address": {
        "street": "Am Neckartor",
        "streetNumber": "2",
        "city": "Stuttgart",
        "zip": "70190",
        "country": "Germany"
      },
      "contact": {
        "email": "e-mobilitaet@enbw.com",
        "web": "www.enbw.com",
        "phone": null,
        "fax": null
      },
      "operator-company-id": 710,
      "operator-logo": "http://operatorlogopath.png",
      "floor-level": null,
      "is-free-charge": false,
      "last-static-change": "2015-01-23T18:54:52+01:00",
      "last-dynamic-change": "2013-02-12T01:43:23+01:00",
      "name": "Am Neckartor",
      "description": null,
      "latitude": 48.786574,
      "longitude": 9.190824,
      "open-hour-notes": [
        {
          "times": [
            "24h"
          ],
          "days": [
            "Mo",
            "Su"
          ]
        }
      ],
      "total-parking": 1,
      "notes": "",
      "is-green-power-available": true,
      "is-plugin-charge": true,
      "is-roofed": false,
      "is-reservable": false,
      "has-dynamic-info": false,
      "is-open-24": true,
      "dynamic-status-summary": null,
      "is-validated": true,
      "is-private": false,
      "owner-company-id": null,
      "service-providers": [
        710
      ],
      "connectors": [
        {
          "id": 11154,
          "status": "Unknown",
          "last-change": "2014-07-01T15:24:28+02:00",

```

```

        "name": "Schuko",
        "speed": "3.7kW",
        "mode": "Model",
        "external-id": "DE*123*1234567",
        "prices": null
    }
}
},
{
    "id": 1169,
    "address": {
        "street": "Südwall",
        "streetNumber": "32",
        "city": "Geldern",
        "zip": "47608",
        "country": "Germany"
    },
    "contact": {
        "email": null,
        "web": "www.stadtwerke-geldern.de",
        "phone": null,
        "fax": null
    },
    "operator-company-id": 715,
    "operator-logo": null,
    "floor-level": null,
    "is-free-charge": false,
    "last-static-change": "2015-01-23T18:54:52+01:00",
    "last-dynamic-change": "2013-02-12T01:49:05+01:00",
    "name": "Marktparkhaus am Südwall",
    "description": null,
    "latitude": 51.516123,
    "longitude": 6.322554,
    "open-hour-notes": [
        {
            "times": [
                "07:30",
                "20:00"
            ],
            "days": [
                "Mo",
                "Fr"
            ]
        },
        {
            "times": [
                "07:30",
                "15:00"
            ],
            "days": [
                "Sa",
                "Sa"
            ]
        }
    ],
    "total-parking": 1,
    "notes": "PARKING CHARGE",
    "is-green-power-available": true,

```

```

    "is-plugin-charge": true,
    "is-roofed": false,
    "is-reservable": false,
    "has-dynamic-info": false,
    "is-open-24": false,
    "dynamic-status-summary": "Available",
    "is-validated": true,
    "is-private": false,
    "owner-company-id": 28,
    "service-providers": [
      715,
      1224
    ],
    "connectors": [
      {
        "id": 11451,
        "status": "Available",
        "last-change": "2014-07-01T15:25:40+02:00",
        "name": "Chademo",
        "speed": "52kW",
        "mode": "Mode4",
        "external-id": "DE*123*E000000002",
        "prices": {
          "starting-fee": "0.00",
          "charging-per-hour": "0.00",
          "parking-per-hour": "1.30",
          "charging-per-kwh": "0.17",
          "currency": "EUR"
        }
      },
      {
        "id": 11452,
        "status": "Occupied",
        "last-change": "2014-07-01T15:25:40+02:00",
        "name": "Type2",
        "speed": "22.2kW",
        "mode": "Mode3",
        "external-id": "DE*123*E000000002",
        "prices": {
          "starting-fee": "0.00",
          "charging-per-hour": "0.00",
          "parking-per-hour": "1.10",
          "charging-per-kwh": "0.36",
          "currency": "EUR"
        }
      }
    ]
  },
  {
    "id": 1770,
    "address": {
      "street": "Torgauer StraÙe",
      "streetNumber": "12",
      "city": "Berlin",
      "zip": "10829",
      "country": "Germany"
    },
    "contact": {

```



```

    "email": null,
    "web": "https://www.rwe-mobility.com",
    "phone": null,
    "fax": null
  },
  "operator-company-id": 715,
  "operator-logo": null,
  "floor-level": null,
  "is-free-charge": false,
  "last-static-change": "2015-01-23T18:54:52+01:00",
  "last-dynamic-change": "2014-12-29T21:48:08+01:00",
  "name": "Torgauer Straße",
  "description": null,
  "latitude": 52.482327,
  "longitude": 13.357278,
  "open-hour-notes": [
    {
      "times": "[24h]",
      "days": [
        "Mo",
        "Su"
      ]
    }
  ],
  "total-parking": 1,
  "notes": "",
  "is-green-power-available": true,
  "is-plugin-charge": true,
  "is-roofed": false,
  "is-reservable": false,
  "has-dynamic-info": true,
  "is-open-24": true,
  "dynamic-status-summary": "Available",
  "is-validated": true,
  "is-private": false,
  "owner-company-id": 28,
  "service-providers": [
    715,
    1224,
    1337,
    1338
  ],
  "connectors": [
    {
      "id": 8613,
      "status": "Available",
      "last-change": "2014-12-29T21:48:08+01:00",
      "name": "Type2",
      "speed": "22.2kW",
      "mode": "Mode3",
      "external-id": null,
      "prices": {
        "starting-fee": "0.00",
        "charging-per-hour": "0.00",
        "parking-per-hour": "1.10",
        "charging-per-kwh": "0.36",
        "currency": "EUR"
      }
    }
  ]

```

```
    },
    {
      "id": 8614,
      "status": "Available",
      "last-change": "2014-12-23T21:22:09+01:00",
      "name": "Type2",
      "speed": "22.2kW",
      "mode": "Mode3",
      "external-id": null,
      "prices": {
        "starting-fee": "0.00",
        "charging-per-hour": "0.00",
        "parking-per-hour": "1.10",
        "charging-per-kwh": "0.36",
        "currency": "EUR"
      }
    }
  ]
},
{
  "id": 2057,
  "address": {
    "street": "Church Row",
    "streetNumber": "23",
    "city": "London",
    "zip": "NW3 6UR",
    "country": "United Kingdom"
  },
  "contact": {
    "email": "membership@sourcelondon.net",
    "web": "https://www.sourcelondon.net/",
    "phone": null,
    "fax": null
  },
  "operator-company-id": 39,
  "operator-logo": null,
  "floor-level": null,
  "is-free-charge": true,
  "last-static-change": "2015-01-23T18:54:52+01:00",
  "last-dynamic-change": "2013-02-19T20:17:41+01:00",
  "name": "Church Row",
  "description": null,
  "latitude": 51.556097,
  "longitude": -0.179109,
  "open-hour-notes": [
    {
      "times": "[24h]",
      "days": [
        "Mo",
        "Su"
      ]
    }
  ]
},
  "total-parking": 1,
  "notes": "",
  "is-green-power-available": false,
  "is-plugin-charge": true,
  "is-roofed": false,
```

```

    "is-reservable": false,
    "has-dynamic-info": false,
    "is-open-24": true,
    "dynamic-status-summary": null,
    "is-validated": true,
    "is-private": false,
    "owner-company-id": null,
    "service-providers": null,
    "connectors": [
      {
        "id": 25443,
        "status": "Unknown",
        "last-change": "2014-08-14T18:00:37+02:00",
        "name": "Type2",
        "speed": "3.7kW",
        "mode": "Mode3",
        "external-id": null,
        "prices": {
          "starting-fee": "0.00",
          "charging-per-hour": "0.00",
          "parking-per-hour": "0.00",
          "charging-per-kwh": "0.00",
          "currency": "EUR"
        }
      },
      {
        "id": 25444,
        "status": "Unknown",
        "last-change": "2014-08-14T18:00:37+02:00",
        "name": "3PinSquare",
        "speed": "3.7kW",
        "mode": "Model",
        "external-id": null,
        "prices": {
          "starting-fee": "0.00",
          "charging-per-hour": "0.00",
          "parking-per-hour": "0.00",
          "charging-per-kwh": "0.00",
          "currency": "EUR"
        }
      }
    ]
  },
  ],
  "companies": [
    {
      "id": 28,
      "name": "RWE",
      "contact": {
        "email": null,
        "web": "https://www.rwe-mobility.com",
        "phone": "0800 2335335",
        "fax": null
      },
      "address": {
        "street": null,
        "streetNumber": null,
        "city": null,

```

```
        "zip": null,
        "country": "United Kingdom"
    },
    "description": "RWE",
    "type": null
},
{
    "id": 39,
    "name": "Source London",
    "contact": {
        "email": "membership@sourcelondon.net",
        "web": "https://www.sourcelondon.net/",
        "phone": "+448458500653",
        "fax": null
    },
    "address": {
        "street": null,
        "streetNumber": null,
        "city": "POOLE",
        "zip": "BH12 9HE",
        "country": "United Kingdom"
    },
    "description": null,
    "type": null
},
{
    "id": 710,
    "name": "ENBW",
    "contact": {
        "email": "e-mobilitaet@enbw.com",
        "web": "www.enbw.com",
        "phone": "+498003629001",
        "fax": null
    },
    "address": {
        "street": null,
        "streetNumber": null,
        "city": null,
        "zip": null,
        "country": "Germany"
    },
    "description": null,
    "type": null
},
{
    "id": 715,
    "name": "RWE",
    "contact": {
        "email": null,
        "web": "https://www.rwe-mobility.com",
        "phone": "+498002255793",
        "fax": null
    },
    "address": {
        "street": null,
        "streetNumber": null,
        "city": null,
        "zip": null,
```

```

        "country": "Germany"
    },
    "description": null,
    "type": null
},
{
    "id": 1224,
    "name": "SMS",
    "contact": {
        "email": null,
        "web": null,
        "phone": null,
        "fax": null
    },
    "address": {
        "street": null,
        "streetNumber": null,
        "city": null,
        "zip": null,
        "country": "Germany"
    },
    "description": null,
    "type": null
},
{
    "id": 1337,
    "name": "PlugSurfing App",
    "contact": {
        "email": "service@plugsurfing.com ",
        "web": "https://www.plugsurfing.com/",
        "phone": null,
        "fax": null
    },
    "address": {
        "street": "Torgauerstr",
        "streetNumber": "12-15",
        "city": "Berlin",
        "zip": "10829",
        "country": "Germany"
    },
    "description": null,
    "type": null
},
{
    "id": 1338,
    "name": "Intercharge QR-Code",
    "contact": {
        "email": "service@plugsurfing.com ",
        "web": "https://www.plugsurfing.com/",
        "phone": null,
        "fax": null
    },
    "address": {
        "street": "Torgauerstr",
        "streetNumber": "12-15",
        "city": "Berlin",
        "zip": "10829",
        "country": "Germany"
    }
}

```

```
        },
        "description": null,
        "type": null
    }
],
"result": {
    "code": 0,
    "message": "Success."
}
}
```

### Request

"station-get-surface" identifies the call as a station-get-surface call.

Define a rectangle on a map by providing two corners:

1. Minimum latitude and longitude
2. Maximum latitude and longitude

### Fields

**min-lat** Minimum latitude of the area you are querying (float).

**max-lat** Maximum latitude of the area you are querying (float).

**min-long** Minimum longitude of the area you are querying (float).

**max-long** Maximum longitude of the area you are querying (float).

**filters** An object to filter the response (object).

Omit any filter you do not want to use in the request.

Available Filters:

**excludes** A list of station IDs to exclude from the result (array of integers).

**company-types** A list of owner types of the charging stations (array of strings).

E.g. "hotel".

**connector-types** A list of connector types to filter for (array of strings).

Allowed types are:

- "Type2"

- "Combo"
- "Chademo"
- "Schuko"
- "Type3"
- "CeeBlue"
- "3PinSquare"
- "Type1"
- "CeeRed"
- "Cee2Poles"
- "Tesla"
- "Scame"
- "Nema5"
- "CeePlus"
- "T13"
- "T15"
- "T23"
- "Marechal"
- "TypeE"

**connector-speeds-greater** Minimum value for charging speed in kW (float).

**connector-speeds-less** Maximum value for charging speed in kw (float).

**operator-ids** List of charging station operator ids (array of integers).

**payable** List of service providers that should be available (array of strings).

Examples are

- "PSA\_EU" for PlugSurfing App payable
- "PSC\_EU" for PlugSurfing Charging Key payable

## Response

### Fields

**stations** An array of charging stations (objects).

**id** The reponder's ID of the station (integer).

Use this ID in the excludes filter (see request) or to get more details. See also *Station Get By IDs*.

**name** The name of the station, human readable (string).

**latitude** Latitude of this station (float).

**longitude** Longitude of this station (float).



**dynamic-status-summary** Whether the station has available connectors (string).

Can be one of:

- "Available"
- "Occupied"
- "Offline"
- null

**owner-type** The type of the company (string or null).

E.g. "hotel".

**last-static-change** The last time the station was updated (string).

The date/time format is RFC3339 (Y-m-d\TH:i:sP).

**connector-statuses** Array of connectors' statuses (id (string): status (string)).

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

## Examples

Request:

```
{
  "station-get-surface": {
    "min-lat": 0,
    "max-lat": 45,
    "min-long": 30,
    "max-long": 40,
    "filters": {
      "excludes": [
        11131
      ],
      "company-types": [
        "hotel"
      ],
      "connector-types": [
        "Type2"
      ],
      "connector-speeds-greater": 3,
      "connector-speeds-less": 100,
      "operator-ids": [
        122,
        32
      ],
      "payable": [
        "app",
```

```

        "rfid"
      ]
    }
  }
}

```

Response:

```

{
  "stations": [
    {
      "id": 1169,
      "name": "Marktparkhaus am Südwall",
      "latitude": 51.516123,
      "longitude": 6.322554,
      "dynamic-status-summary": null,
      "owner-type": null,
      "last-static-change": "2017-01-13T18:07:23+01:00",
      "connector-statuses": {
        "165946": "Available",
        "165947": "Available"
      }
    },
    {
      "id": 1622,
      "name": "Markt",
      "latitude": 51.51599,
      "longitude": 6.322551,
      "dynamic-status-summary": null,
      "owner-type": null,
      "last-static-change": "2017-01-13T18:07:23+01:00",
      "connector-statuses": {
        "142867": "Unknown"
      }
    }
  ],
  "result": {
    "code": 0,
    "message": "Success."
  }
}

```

---

## Station Get Usage

---

### Request

"station-get-usage" identifies the call as a station-get-usage call.

### Fields

**station-id** A station ID (integer).

### Response

#### Fields

Optional fields may be omitted or have the value `null`.

**station-usage** This field contains the station usage data (object).

**last-7-days-usage-count** Sessions within the last 7 days (integer).

**last-30-days-usage-count** Sessions within the last 30 days (integer).

**last-station-usage-date** Date of the last usage of the station (string).

        The date/time format is RFC3339 (Y-m-d\TH:i:sP).

### HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**181** EVSE not found

## Examples

Request:

```
{
  "station-get-usage": {
    "station-id": 1770
  }
}
```

Response:

```
{
  "station-usage": {
    "last-7-days-usage-count": 2,
    "last-30-days-usage-count": 4,
    "last-station-usage-date": "2017-04-12T19:10:02+02:00"
  }
}
```

---

## Station Post

---

**Warning:** It is not allowed to change the connectors of an existing station.

**Warning:** The station's connectors' IDs may not be changed once it is created.

You have to send one request for each station. It is not possible to send multiple stations with one request.

---

**Note:** To ensure very high data quality, PlugSurfing manually curates all charging station and connector data. This means that data on the PlugSurfing platform may not correspond 100% to the data you push.

Furthermore, PlugSurfing groups stations to simplify access from clients like the PlugSurfing app. As a result, multiple stations you push may show as a single station on the PlugSurfing platform, with all connectors from all individual stations associated to the grouped station.

---

## Request

"station-post" identifies the call as a station-post call.

## Fields

---

**Note:** PlugSurfing uses the station's operator's name to display a name of a charging station in the mobile app or on the web. In case an operator name is missing, PlugSurfing displays the station name instead.

---

**station** An object that holds the station data. Fields of a station are:

**id** An identifier that uniquely identifies the station (string).

---

**Important:** Every station id must be globally unique.

---

**name** The name is shown to the EV driver (string).

**description (optional)** A more detailed description of the station (string).

**latitude** Latitude of the station (float).

**longitude** Longitude of the station (float).

**address** The address helps EV drivers locate the station (object).

**street** string

**street-number** string

**city** string

**zip** string

**country** string

Format: international two-letter codes in accordance with [ISO 3166-1 alpha-2](#).

**contact** object

**phone** string

**fax (optional)** string

**website (optional)** string

**email (optional)** string

**cpo-id** The EVSE identifier, e.g. DE\*8PS, of the CPO of the charging station (string).

**is-open-24** Whether or not the station is always accessible (24 hours per day) (boolean).

**connectors** array of objects

All the connectors belonging to this station:

**id** The EVSE ID of the connector (string).

See also [EVSE](#).

---

**Important:** The id of every connector must be globally unique.

---

**name** The name of the type of connector (string).

Allowed types are:

- "Type2"
- "Combo"
- "Chademo"
- "Schuko"
- "Type3"
- "CeeBlue"

- "3PinSquare"
- "Type1"
- "CeeRed"
- "Cee2Poles"
- "Tesla"
- "Scame"
- "Nema5"
- "CeePlus"
- "T13"
- "T15"
- "T23"
- "Marechal"
- "TypeE"

If your type is missing, please do not hesitate to contact PlugSurfing.

**speed** Speed in kW (float).

**open-hour-notes (optional)** An array of objects containing certain opening periods:

**times** Opening and closing time (array of strings).

**days** Weekdays when the interval starts and ends (array of two strings).

Both are the same if it is for one specific day only.

Example:

```
{
  "open-hour-notes": [
    {
      "times": [
        "07:30",
        "19:00"
      ],
      "days": [
        "Mo",
        "Fr"
      ]
    },
    {
      "times": [
        "09:00",
        "15:00"
      ],
      "days": [
        "Sa",
        "Sa"
      ]
    }
  ]
}
```

This example means the following: For the interval Monday to Friday, the station is open from 07:30 to 19:00. On Saturday, the station is open from 09:00 to 15:00.

**notes (optional)** Additional notes, for example how to find the station (string).

**is-reservable (optional)** boolean

**floor-level (optional)** On which floor the station is located, for example in a parking house (integer).

**is-free-charge (optional)** Whether charging can be done without cost (boolean).

**total-parking (optional)** The number of parking spots that are available at the station (integer).

**is-green-power-available (optional)** boolean

**is-plugin-charge (optional)** Whether or not a user can authorize with the proprietary “Plugin-Charge” method (boolean).

**is-roofed (optional)** Whether the station is under a roof, for example in a parking garage (boolean).

**is-private (optional)** Whether the station is privately owned (boolean).

This has multiple implications depending on the connected partner and the station won’t show up everywhere on their platforms. For details, please contact the connected partner.

**deleted** Soft delete the station and its related connectors (boolean).

**partner-identifier** The partner identifier of the partner that shall be associated with this station. See also *partner identifier*

## Response

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**211** Invalid partner identifier

## Examples

Request:

```
{
  "station-post": {
    "station": {
      "id": "abcdef-12345",
      "name": "test",
      "description": "Nice station!",
      "latitude": 1.123,
      "longitude": 2.345,
      "address": {
        "street": "streetname",
        "street-number": "123a",
```



```

    "city": "Berlin",
    "zip": "10243",
    "country": "DE"
  },
  "contact": {
    "phone": "+49 30 8122321",
    "fax": "+49 30 8122322",
    "web": "www.example.com",
    "email": "contact@example.com"
  },
  "cpo-id": "DE*8PS",
  "is-open-24": false,
  "connectors": [
    {
      "id": "DE*8PS*E123456",
      "name": "Schuko",
      "speed": 3.7
    },
    {
      "id": "DE*8PS*E123457",
      "name": "Type2",
      "speed": 11.1
    }
  ],
  "open-hour-notes": [
    {
      "times": [
        "07:30",
        "19:00"
      ],
      "days": [
        "Mo",
        "Fr"
      ]
    },
    {
      "times": [
        "09:00",
        "15:00"
      ],
      "days": [
        "Sa",
        "Sa"
      ]
    }
  ],
  "notes": "Additional info.",
  "is-reservable": false,
  "floor-level": 1,
  "is-free-charge": false,
  "total-parking": 2,
  "is-green-power-available": false,
  "is-plugin-charge": false,
  "is-roofed": false,
  "is-private": false,
  "deleted": true
},
"partner-identifier": "1"

```

```
}  
}
```

#### Response

```
{  
  "result": {  
    "code": 0,  
    "message": "Success."  
  }  
}
```

---

## User Add Credit Card

---

### Request

"user-add-creditcard" identifies the call as a user-add-creditcard call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**encrypted-data** Encrypted credit card data (string)

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

- 0** Success
- 140** Authentication failed: No positive authentication response
- 144** Authentication failed: Email does not exist
- 145** Authentication failed: User token not valid
- 805** This user is not allowed to use this method
- 870** Credit card error

## Examples

Request:

```
{
  "user-add-creditcard": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "87d4e3085af04671834ebeb127df33bf"
    },
    "encrypted-data":
    ↪ "zpTii0RDoCl7SzDF9dVIKhNFI0rzpTii0RDoCl7SzDF9dVIKhNFI0rzpTii0RDoCl7SzDF9dVIKhNFI0r"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-add-payment" identifies the call as a user-add-payment call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**payment-type** The type of payment that shall be added (string).

Can be one of (directEbanking = Sofortueberweisung):

- "paypal"
- "directEbanking"
- "ideal"

## Response

### Fields

**url** The JSON and UTF encoded URL for the request (string)

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**800** Payment system error

**805** This user is not allowed to use this method

## Examples

Request:

```
{
  "user-add-payment": {
    "payment-type": "paypal",
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "87d4e3085af04671834ebeb127df33bf"
    }
  }
}
```

Response:

```
{
  "user-add-payment": {
    "url": "https://test.adyen.com/hpp/details.shtml?&paymentAmount=100&
    ↪currencyCode=EUR&shipBeforeDate=2015-03-01&
    ↪merchantReference=Authorization+youridentifier&skinCode=GfUFVL5L&
    ↪merchantAccount=PlugSurfing&sessionValidity=2015-02-27T14%3A47%3A28%2B01%3A00&
    ↪shopperEmail=customer%40gmail.com&shopperReference=youridentifier&allowedMethods=&
    ↪blockedMethods=&offset=&recurringContract=RECURRING&
    ↪orderData=H4sIAAAAAAAAAAwvIKU0PLi1Ky8xLBwBbAAADCwAAAA%3D%3D&countryCode=DE&
    ↪brandCode=paypal&merchantSig=2LUxxOwNdXV9nnAAAAJ4J%2FE4V8%3D"
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-add-rfid" identifies the call as a user-add-rfid call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**rfid** The RFID (UID) that identifies the charging key (string).

---

#### Important:

- The format must be a hexadecimal string.
- An RFID must have a length of 8, 14 or 20 characters. If necessary, the RFID must be zero-padded on the left.
- It should be read from left to right using big-endian format.

- All characters must be upper-case.
- 

## Response

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

**190** EVCO ID error

## Examples

Request:

```
{
  "user-add-rfid": {
    "user": {
      "identifier-type": "username",
      "identifier": "john",
      "token": "b3853b6d910849f3b4392555b8acb984"
    },
    "rfid": "12345678ABCDEF"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



### Request

"user-block-rfid" identifies the call as a user-block-rfid call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**rfid** The RFID (UID) of the user that should be blocked (string).

---

#### Important:

- An RFID must have a length of 8, 14 or 20 characters. If necessary, the RFID must be zero-padded on the left.
- It should be read from left to right using big-endian format.
- All characters must be upper-case.

## Response

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

**190** EVCO ID error

**191** EVCO ID not found

## Examples

Request:

```
{
  "user-block-rfid": {
    "user": {
      "identifier-type": "username",
      "identifier": "john",
      "token": "b3853b6d910849f3b4392555b8acb984"
    },
    "rfid": "12345678ABCDEF"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

---

## User Change Password

---

### Request

"user-change-password" identifies the call as a user-change-password call.

### Fields

**username** This field identifies the customer by username or email (string).

**current-password** Current password of the customer (string).

**new-password** New password of the customer (string).

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

#### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

## Examples

Request:

```
{
  "user-change-password": {
    "username": "myusername",
    "current-password": "oldpassword",
    "new-password": "mynewpassword"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

---

## User Charging Key Activate

---

### Request

"user-charging-key-activate" identifies the call as a user-charging-key-activate call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**evco-id** The EVCO ID that identifies the charging key (string).

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

- 0** Success
- 140** Authentication failed: No positive authentication response
- 144** Authentication failed: Email does not exist
- 145** Authentication failed: User token not valid
- 190** EVCO ID error

## Examples

Request:

```
{
  "user-charging-key-activate": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "yourtoken"
    },
    "evco-id": "yourevcoid"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-get-bills" identifies the call as a user-get-bills call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**date-interval** This field sets the date interval for which bills should be returned (object).

**start** Date/Time as string in format Y-m-dTH:i:sP. E.g. 2015-03-04T13:32:12+01:00.

**end** Date/Time as string in format Y-m-dTH:i:sP. E.g. 2015-03-06T00:00:00+01:00.

## Response

### Fields

**bills** This field contains the bills for the requested date interval (array).

**external-session-ids** List of session ID's for the invoice (array).

**bill** URL with link to the invoice (string).

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-get-bills": {
    "user": {
      "identifier-type": "username",
      "identifier": "some_user",
      "token": "b369f99e82fa097ba9cff8658c74e47c"
    },
    "date-interval": {
      "start": "2015-03-04T13:32:12+01:00",
      "end": "2015-03-06T00:00:00+01:00"
    }
  }
}
```

Response:

```
{
  "user": {
    "bills": [
      {
        "external-session-ids": [
          "6a2f25f6-0a88-1293-5ab0-0e3da7abcdef"
        ],
        "bill": "https://api.plugsurfing.com/uploads/bills/Invoice-201503051-
↪10000.pdf"
      }
    ]
  }
}
```



```
    {
      "external-session-ids": [
        "6a2f25f6-a82c-1234-9090-0e3da7abcde1",
        "6a2f25f6-927f-6578-1432-8394caabcde2"
      ],
      "bill": "https://api.plugsurfing.com/uploads/bills/Invoice-201503052-
↪10000.pdf"
    }
  ],
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



---

## User Get Charging Keys

---

### Request

"user-get-charging-keys" identifies the call as a user-get-charging-keys call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

### Response

#### Fields

**charging-keys** An array of user charging keys (array of objects).

**uid** Charging key UID (string).

**evco-id** Charging key EVCO ID (string).

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-get-charging-keys": {
    "user": {
      "identifier-type": "username",
      "identifier": "some_user",
      "token": "b369f99e82fa097ba9cff8658c74e47c"
    }
  }
}
```

Response:

```
{
  "user": {
    "charging-keys": [
      {
        "uid": "ABCDABCD",
        "evco-id": "DE*8PS*C12345"
      },
      {
        "uid": "ABCDABCF",
        "evco-id": "DE*8PS*C12346"
      }
    ]
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-get-details" identifies the call as a user-get-details call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

### Response

#### Fields

**user** This field contains the user (object).

**first-name** First name of a user, can be null (string).

**last-name** Last name of a user, can be null (string).

**address** The address where the user lives. Can be null (object).

**street** Street name (string).

**street-number** Street number (string).

**zip** Zipcode for the address (string).

**city** Name of the City (string).

**country** Country, mandatory (string).

Please use international two letter codes (ISO 3166-1 alpha-2).

**billing-name** If the name on the bill should not be first and lastname. Can be null (string).

**billing-address** The address where the user lives. Can be null (object).

**street** Street name (string).

**street-number** Street number (string).

**zip** Zipcode for the address (string).

**city** Name of the City (string).

**country** Country, mandatory (string).

Please use international two letter codes (ISO 3166-1 alpha-2).

**locale** Locale of the user (string).

Two lower case letters. E.g. en for English or de for German. Can be null.

**vat** The VAT number is validated, can be null (string).

**social-security-number** Social Security Number, can be null (string).

**evco-id** The EVCO-ID of the Customer, can be null (string).

**phone** The phone of the Customer, can be null (string).

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-get-details": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "87d4e3085af04671834eb127df33bf"
    }
  }
}
```

Response:

```
{
  "user": {
    "first-name": "Firstname",
    "last-name": "Lastname",
    "address": {
      "street": "Warschauer Str.",
      "street-number": "1",
      "city": "Berlin",
      "zip": "10247",
      "country": "Germany"
    },
    "billing-name": "PlugSurfing GmbH",
    "billing-address": {
      "street": "Torgauer Str.",
      "street-number": "12-15",
      "city": "Berlin",
      "zip": "10829",
      "country": "Germany"
    },
    "locale": "de",
    "vat": "DE123456",
    "social-security-number": "SocialSecNumber",
    "evco-id": "DE*8PS*156456730*9",
    "phone": "+49 151 84512991"
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```





---

## User Get Recent Sessions

---

### Request

"user-get-recent-sessions" identifies the call as a user-get-recent-sessions call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

### Response

#### Fields

Optional fields may be omitted or have the value `null`.

**sessions** An array of sessions (array of objects).

**request-sent (optional)** The date/time at which the session was requested by the user (string; format: "2014-06-23T18:32:49+02:00").

**connector-id (optional)** Internal database ID of the related connector (integer).

**charging-station-id (optional)** Internal database ID of the related charging station (integer).

**session-interval** Start and end time of the session (object).

The session interval is the time between connecting the car to the station and disconnecting.

**start (optional)** The date/time at which the session was started (string; format: "2014-06-23T18:32:49+02:00").

**end (optional)** The date/time at which the session was stopped (string; format: "2014-06-23T18:32:49+02:00").

**charging-interval** Start and end time of charging (object).

The charging interval is the time between the first and the last time that energy was transferred to the car.

**start (optional)** The date/time at which charging started (string; format: "2014-06-23T18:32:49+02:00").

**end (optional)** The date/time at which charging stopped (string; format: "2014-06-23T18:32:49+02:00").

**energy-consumed (optional)** The consumed energy in kWh (float).

**cost** The total cost of the session (string).

**currency** Currency of the cost (string).

**external-session-id (optional)** The session ID at the CPO (string).

**address (optional)** The address where the session took place (string).

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-get-recent-sessions": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
    }
  }
}
```

```

    "token": "87d4e3085af04671834ebeb127df33bf"
  }
}

```

Response:

```

{
  "user": {
    "sessions": [
      {
        "request-sent": "2014-06-23T18:32:49+02:00",
        "connector-id": 9835,
        "charging-station-id": 19018,
        "session-interval": {
          "start": "2014-06-23T16:32:22+02:00",
          "end": "2014-06-23T17:42:47+02:00"
        },
        "charging-interval": {
          "start": "2014-06-23T16:32:28+02:00",
          "end": "2014-06-23T17:30:21+02:00"
        },
        "energy-consumed": 3,
        "cost": "3.76",
        "currency": "EUR",
        "external-session-id": "abc-def",
        "address": "Tempelhofer Ufer 17, 10963 Berlin, Germany"
      },
      {
        "request-sent": "2014-06-23T18:40:50+02:00",
        "connector-id": null,
        "charging-station-id": null,
        "session-interval": {
          "start": "2014-06-23T16:38:18+02:00",
          "end": "2014-06-23T16:40:45+02:00"
        },
        "charging-interval": {
          "start": null,
          "end": null
        },
        "energy-consumed": 0.053,
        "cost": "2.03",
        "currency": "EUR",
        "external-session-id": "abc-def",
        "address": "Tempelhofer Ufer 17, 10963 Berlin, Germany"
      }
    ]
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}

```



---

## User Get Recent Stations

---

### Request

"user-get-recent-stations" identifies the call as a user-get-recent-stations call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**limit** Limit the response to this number of stations (integer).

### Response

#### Fields

**stations** An array of stations (array of objects).

**id** The responder's ID of the station (integer).

Use this ID to get more details. See also *Station Get By IDs*.

**name** The name of the station, human readable (string).

**latitude** Latitude of this station (float).

**longitude** Longitude of this station (float).

**dynamic-status-summary** Whether the station has available connectors (string).

Can be one of:

- "Available"
- "Occupied"
- "Offline"
- null

**owner-type** The type of the company (string or null).

E.g. "hotel".

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-get-recent-stations": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "87d4e3085af04671834eb127df33bf"
    },
    "limit": 2
  }
}
```

Response:

```
{
  "user": {
    "stations": [
      [
        {
          "id": 1247,
          "name": "Vattenfall Ladestation",
          "latitude": 52.52119,
          "longitude": 13.32143,
          "dynamic-status-summary": "Available",
          "owner-type": null
        },
        {
          "id": 1248,
          "name": "Hotel Station",
          "latitude": 52.82119,
          "longitude": 12.12143,
          "dynamic-status-summary": "Occupied",
          "owner-type": "hotel"
        }
      ]
    ]
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```





### Request

"user-logout" identifies the call as a user-logout call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-logout": {
    "user": {
      "identifier-type": "username",
      "identifier": "youridentifier",
      "token": "87d4e3085af04671834ebeb127df33bf"
    }
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-post-details" identifies the call as user-post-details call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**details** This field contains the details of a user (object)

**first-name** First name of a user, can be null (string).

**last-name** Last name of a user, can be null (string).

**address** The address where the user lives. Can be null (object).

**street** Street name (string).

**street-number** number (string).

**zip** Zipcode for the address (string).

**city** Name of the City (string).

**country** Country, mandatory (string).

Please use international two letter codes (ISO 3166-1 alpha-2).

**locale** Defining the locale of the user (string).

Two lower case letters. E.g. en for English or de for German. Can be null.

**vat** The VAT number is validated, can be null (string).

**social-security-number** Social security number, can be null (string).

**billing-name** If the name on the bill should not be first and lastname. Can be null (string).

**billing-address** In case the address on the bill shall be different from the address, it can be specified. Can be null (object).

**street** Street name (string).

**street-number** Street number (string).

**zip** Zipcode for the address (string).

**city** Name of the City (string).

**country** Country, mandatory (string).

Please use international two letter codes (ISO 3166-1 alpha-2).

**phone** The phone of the Customer, can be null (string).

## Response

### Fields

**reason** An optional response field. If present, it is a string stating the reason for an error.

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

**230** Invalid request format

## Examples

Request:

```
{
  "user-post-details": {
    "user": {
      "identifier-type": "username",
      "identifier": "iAmUser",
      "token": "abababa"
    },
    "details": {
      "first-name": "Firstname",
      "last-name": "Lastname",
      "address": {
        "street": "Torgauer Str.",
        "street-number": "12 - 15",
        "zip": "10829",
        "city": "Berlin",
        "country": "DE"
      },
      "locale": "de",
      "vat": "DE1234567",
      "social-security-number": null,
      "billing-name": "PlugSurfing GmbH",
      "billing-address": {
        "street": "Torgauer Str.",
        "street-number": "12 - 15",
        "zip": "10829",
        "city": "Berlin",
        "country": "DE"
      },
      "phone": "+49 151 84512991"
    }
  }
}
```

Response (success true):

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

Response (success false):

```
{
  "user-post-details": {
    "reason": "Could not validate VAT number: DE1234567"
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



### Request

---

**Note:** The field “language” is optional.

---

### Fields

**username** The desired username of the new user (string).

**email** The email of the new user (string).

**password** The plain password of the new user (string).

**language** The locale of the new user (string). Expected is a lowercase, two-character string in accordance with [ISO 639-1:2002](#).

### Response

#### Fields

**token** A token to authenticate the user in future requests (string).

Keep this token. When you make future requests where the user needs to be authenticated, you supply this token with the request.

**evco-id** The EVCO ID of the user (string). See also *EVCO ID*

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**143** Authentication failed: Email already exists

## Examples

Request:

```
{
  "user-register": {
    "username": "user",
    "email": "name@mail.com",
    "password": "plain-password",
    "language": "en"
  }
}
```

Response:

```
{
  "user": {
    "token": "e2af72d7a9084431ab0b1a5c42df7745",
    "evco-id": "DE*8PS*123456*7"
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```



### Request

"user-reset" identifies the call as a user-reset call.

### Fields

**identifier** The username or the email of the user (string).

### Response

#### HTTP Status codes

**200 OK** The request was processed successfully.

#### Result codes

**0** Success

**144** Authentication failed: Email does not exist

### Examples

Request:

```
{
  "user-reset": {
    "identifier": "usernameoremail"
  }
}
```

Response:

```
{
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"user-unblock-rfid" identifies the call as a user-unblock-rfid call.

### Fields

**user** This field identifies the customer (object).

**identifier-type** How to identify the user (string).

The identifier-type can be one of:

- "evco-id"
- "rfid"
- "username"

**identifier** The identifier is something that uniquely identifies the customer, depending on the identifier-type (string).

**token (optional)** A token can be used to authenticate the user (string).

For example: if the identifier type is username and the identifier is the user's username, then token is used for authentication instead of a password.

**rfid** The RFID (UID) of the user that should be blocked (string).

---

#### Important:

- An RFID must have a length of 8, 14 or 20 characters. If necessary, the RFID must be zero-padded on the left.
- It should be read from left to right using big-endian format.
- All characters must be upper-case.

## Response

### HTTP Status codes

**200 OK** The request was processed successfully.

### Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

**190** EVCO ID error

**191** EVCO ID not found

## Examples

Request:

```
{
  "user-unblock-rfid": {
    "user": {
      "identifier-type": "username",
      "identifier": "john",
      "token": "b3853b6d910849f3b4392555b8acb984"
    },
    "rfid": "12345678ABCDEF"
  }
}
```

Response:

```
{
  "user-unblock-rfid": {
    "success": true
  }
}
```

### Request

#### Fields

---

**Note:** The field “language” is optional.

---

**username** The username of the user (string).

**password** The plain password or token of the user (string).

**is-token** Defines whether the supplied password is actually a token (boolean).

If `is-token` is `true`, the password is interpreted as a token of the user. Otherwise, the password is interpreted as the plain password of the user.

**language** The locale of the user (string).

Expected is a lowercase, two-character string in accordance with [ISO 639-1:2002](#).

### Response

#### Fields

**token** A token to authenticate the user in future requests (string).

Keep this token. When you make future requests where the user needs to be authenticated, you supply this token with the request.

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

**140** Authentication failed: No positive authentication response

**141** Authentication failed: Invalid email or password

**144** Authentication failed: Email does not exist

**145** Authentication failed: User token not valid

## Examples

Request:

```
{
  "user-verify": {
    "username": "your-user",
    "password": "mypassword",
    "is-token": false,
    "language": "en"
  }
}
```

Response:

```
{
  "user": {
    "token": "e2af72d7a9084431ab0b1a5c42df7745"
  },
  "result": {
    "code": 0,
    "message": "Success."
  }
}
```

### Request

"vehicle-get-all" identifies the call as a vehicle-get-all call.

### Fields

There are no fields in the request.

### Response

#### Fields

**vehicles** An array of objects.

**id** The internal DB ID of the vehicle (integer).

**brand** The maker of the vehicle (string).

**model** The model of the vehicle (string).

**max-speed** The maximum charging speed of the vehicle in kW (float).

**connector-types** An array of connector types the vehicle supports (strings).

Types are of:

- "Type2"
- "Combo"
- "Chademo"
- "Schuko"

- "Type3"
- "CeeBlue"
- "3PinSquare"
- "Type1"
- "CeeRed"
- "Cee2Poles"
- "Tesla"
- "Scame"
- "Nema5"
- "CeePlus"
- "T13"
- "T15"
- "T23"
- "Marechal"

## HTTP Status codes

**200 OK** The request was processed successfully.

## Result codes

**0** Success

## Examples

Request:

```
{
  "vehicle-get-all": {}
}
```

Response:

```
{
  "vehicle-get-all": {
    "vehicles": [
      {
        "id": 1,
        "brand": "BMW",
        "model": "i3",
        "max-speed": 3.7,
        "connector-types": [
          "Type3"
        ]
      },
      {

```



```
        "id": 2,  
        "brand": "Renault",  
        "model": "Twizzy",  
        "max-speed": 45,  
        "connector-types": [  
            "Schuko"  
        ]  
    }  
]  
,  
"result": {  
    "code": 0,  
    "message": "Success."  
}
```