ECISS Contributions for OCPI v2.2 & v3.0

ECISS DELIVERABLE WP6 - VERSION 1.2

ECISS

From electric vehicle to smart society

a TKI Urban Energy project

ECISS: E-MOBILITY COMMUNICATION & INFORMATION SYSTEM STRUCTURE
NKL Nederland | Vondellaan 162, 3521 GH Utrecht | www.nklnederland.nl/eciss/

Contents

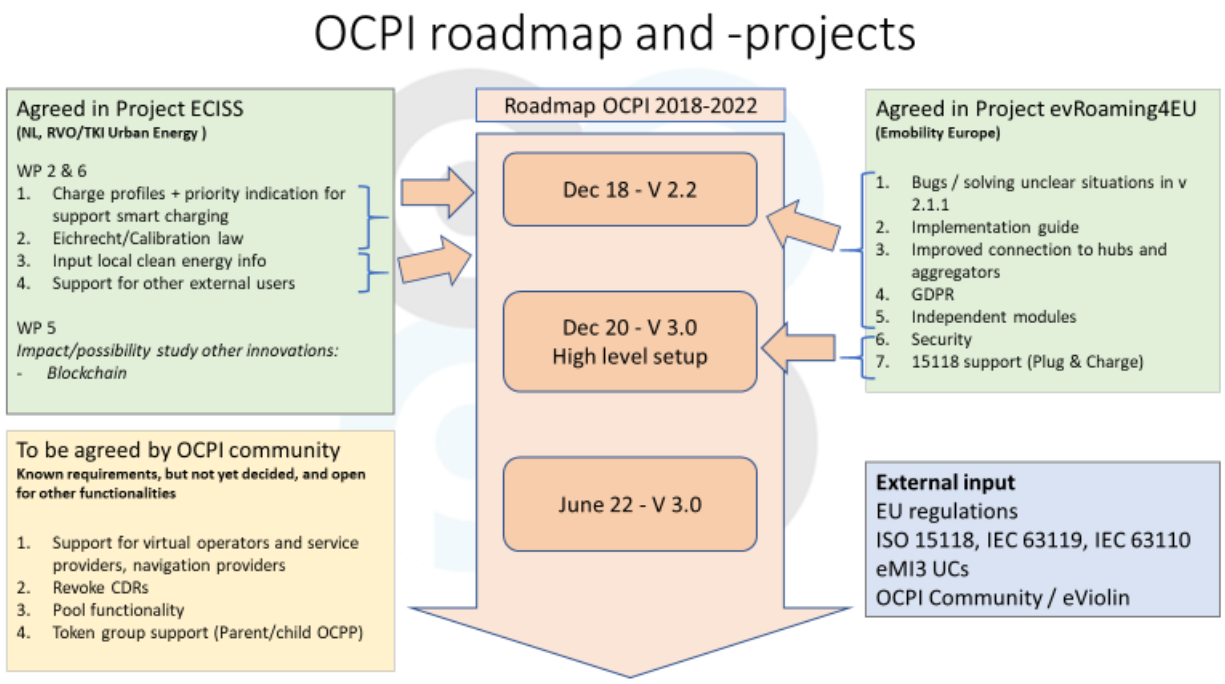
Introduction

1. Objectives related to OCPI protocol

From Electric Vehicle to a Smart Society is strongly related to having the right technologies available. Because of that a main objective of ECISS is improve the OCPI protocol to achieve better national and international connection to market demands and support a faster roll out with extended services for EV drivers in an interoperable way.

1.1 Roadmap

The ECISS project is strongly connected to the OCPI roadmap as shown below.



OCPI is developed via three different ways:

1. Different specific projects. Currently:
    a. ECISS project
    b. evRoaming4EU project
2. Input directly from the OCPI community
3. Input from external organizations e.g.: ISO, IEC, eMI3, eViolin, etc

1.2 Supporting functionalities

In the ECISS proposal and project two main issues are mentioned regarding the objectives:

On national level one of the biggest issues is support for Smart Charging. Not only on technical level but also on functional and usability level.

On international level the main topic was support of Calibration law, which is also referred to as “Eichrecht”, referring to the Germany implementation of the Calibration law for electric mobility.

There are two other supported functionalities that are part of ECISS:

- Input for local green energy usage
- Support for other roles

These last two functionalities have been integrated in OCPI via ECISS input, but not further described in this document.

1.3 Improved efficiency via OCPI v3.0

To improve the efficiency of OCPI, the ECISS project also contributed to the first initial setup of OCPI v3.0 which is based on a radical architectural change and improvement. During the project the first high level overview has been created.

1.4 Link to OCPI specs and documentation

All OCPI specs and documentation of OCPI v2.2 can be found at: www.ocpi-protocol.org or www.evroaming.org.

OCPI v3.0 information is not yet public available. Until final publication of the total specs and documents this is only for project partners and OCPI community members that singed the IPR/CLA policies for OCPI. Upon request at the board of the EVRoaming foundation this information can be shown to others.

2. Smart charging support

Supporting smart charging is based on the Use Cases defined in the ECISS project. It is added in OCPI as separate module. Officially as part of v2.2. but also usable with v2.1.1.

The OCPI functionality in itself is not Smart Charging functionality but can be used for requests to adjust the charging profiles of a running charging session. It can also be used for other charging demands and requirements. Because of that it was decided to name it: Charging profiles that can be used for supporting Smart Charging.

2.1 Description

With the ChargingProfiles module, parties with the role SCSP - Smart Charging Service Provider (can also be used by MSPs) can send (Smart) Charging Profiles requests to a Location/EVSE via the CPO/CSO. It is also possible to request the 'ActiveChargingProfile' from a Location/EVSE.

The ActiveChargingProfile is the charging profile as calculated by the EVSE. It is the result of the calculation of all smart charging inputs present in the EVSE, also Local Limits might be considered.

The ChargingProfile is similar to the concept of Charging Profiles in OCPP but exposes this functionality to third parties. These objects and the accompanying interfaces make certain abstractions that make them more suitable for energy parties to signal their intent. The data structures are based on OCPP 1.6 and 2.0 to make conversion of messages between OCPI and OCPP easy.

Charging Profiles set via this module are no guarantee that the EV will charge with the exact given limit, it is a maximum limit, not a target. A lot of factors influence the charging speed. The EV might not take the amount of energy that the EVSE is willing to provide to it, the battery might be too warm or almost full. A single-phase cable might be used on a three phase Charge Point. There can be local energy limits (load balancing between EVSEs on a relatively small energy connection to a group of EVSEs) that might limit the energy offered by the EVSE to the EV even further.

ChargingProfile can be created by the owner of a Token on Sessions that belong to that token. If another party sends a ChargingProfile and the CPO/CSO has no contract that allows that party to set profiles on sessions, the CPO/CSO is allowed to reject such profiles.

This module can be used by the eMSP, but can also be used by another party that provide "Smart Charging Services" (Smart Charging Service Provider (SCSP) / Aggregator / Energy Service Broker etc.) These SCSPs then depend on the CPO sending session information to them. They need to know which session is ongoing to be able to influence it. If a SCSP uses this module, read eMSP as SCSP.

OCPI provides the means for SCSPs to do this. Parties doing this have to adhere to local privacy laws, have to have setup contracts etc. Local laws might oblige explicit consent from the driver etc.

2.2 Smart Charging Topologies

There are different Smart Charging Topologies possible. Which topology can be used depends on the contracts between different parties.

Care must be taken to prevent mixing the different topologies. When multiple parties start sending Charging Profiles, the resulting charging speed might be unpredictable. In case of OCPP Charge Points, the result will be the minimum of all the Charging Profiles, resulting in a slower than needed charging speed.

2.2.1 The MSP generates Charging Profiles

The most straight forward topology, the MSP (or eMSP) generates ChargingProfiles for its own customers, no SCSP is involved. The eMSP 'owns' the customer, so if the eMSP knows that its customer agrees with the eMSP manipulating the charging speed, the eMSP is free to do this.

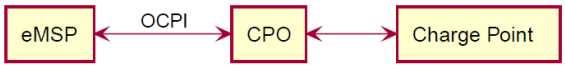


Figure 1: Smart Charging Topology: The eMSP generates ChargingProfiles.

Interface	**Role**
Sender	eMSP
Receiver	CPO

2.2.2 The eMSP delegated Smart Charing to SCSP

In the topology, the eMSP has delegated the generation of ChargingProfiles to a SCSP. For this, the eMSP and SCSP have agreed to use OCPI as the interface.

The eMSP 'owns' the customer, so if the eMSP knows that its customer agrees with the eMSP manipulating the charging speed, the eMSP is free to do this. The eMSP can forward OCPI Session Objects to the SCSP. the SCSP can act on the received/updated Session Objects, by sending Charging Profile commands via the eMSP to the CPO.

The eMSP and SCSP must consider that they have to oblige to local privacy laws when exchanging information about eMSPs customers.

From the CPO point of view, this topology is like the one above, the CPO will not know the difference.

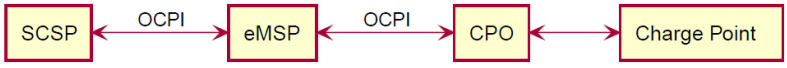


Figure 2. Smart Charging Topology: The eMSP generates ChargingProfiles.

Connection	**Interface**	**Role**
SCSP - eMSP	Sender	SCSP
SCSP - eMSP	Receiver	eMSP
eMSP - CPO	Sender	eMSP
eMSP - CPO	Receiver	CPO

2.2.3 The CPO delegated Smart Charging to SCSP

In this topology, the CPO has delegated the generation of ChargingProfiles to a SCSP. For this, the CPO and SCSP have agreed to use OCPI as the interface.

The CPO 'owns' the EVSE on which charging happens. As the CPO does not 'own' the customers, the CPO needs to make sure the EV driver knows that the charging speed might not be the maximum the driver has expected, this could be something as simple as a sticker on the Charge Point, or might even be part of the tariff text.

The CPO might generate ChargingProfiles themselves, but as OCPI is then not used this is not part of this document.

The CPO can forward OCPI Session Objects to the SCSP. the SCSP can act on the received/updated Session Objects, by sending Charging Profile commands to the CPO.

The CPO and SCSP must take into account that they have to oblige to local privacy laws when exchanging information about eMSPs customers.

In this topology, the eMSP is not aware that the CPO is using OCPI to receive Charging Profiles from the SCSP.

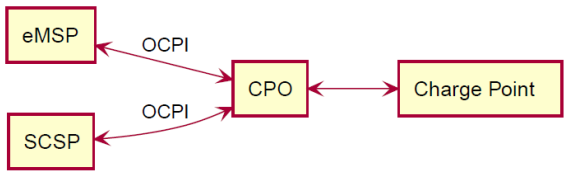


Figure 3. Smart Charging Topology: The eMSP generates ChargingProfiles.

Interface	**Role**
Sender	SCSP
Receiver	CPO

2.3 Use Cases

This module is designed to support the following use cases, for all the above mentioned topologies.

- The eMSP/SCSP sends/updates a ChargingProfile to manipulate an ongoing charging session.
- The eMSP/SCSP request to remove the set ChargingProfile from an ongoing charging session.
- The eMSP/SCSP request the ActiveChargingProfile for an ongoing charging session.
- The CPO updates the eMSP/SCSP of changes to an ActiveChargingProfile.

2.4 Flow

The ChargingProfile creation is a request to activate a charging profile on a running charging session. Most Charge Points are hooked up to the internet via a relative slow wireless connection. To prevent long blocking calls, the ChargingProfile module is designed to work asynchronously. (like the Commands module).

The Sender (Typically SCSP) sends a request to a Receiver (Typically CPO), via the Receiver interface. The Receiver checks if it can send the request to a Charge Point and will respond to the request with a status, indicating if the request can be sent to a Charge Point.

The Receiver sends the requested command (via another protocol, for example: OCPP) to a Charge Point. The Charge Point will respond if it understands the command and will try to execute the command. This response does not always mean that the ChargingProfile will be executed. The CPO will forward the result in a new POST request to the Sender (Typically SCSP) ChargingProfile interface.

The Sender (typically SCSP) can send the Charging Profile to the EVSE via the CPO by using the CPO PUT method for an ongoing session. The Sender can request the current profile the EVSE has calculated, based on different inputs, and is planned to be used for the ongoing session by calling the CPO GET method. The Sender has the ability to remove the Charging Profile for the session by calling the CPO DELETE method.

When the Sender has (at least once) successfully sent a Charging Profile for an ongoing charging session, the Receiver (typically CPO) SHALL keep the Sender updated with changes to the ActiveChargingProfile of that Session. If the Receiver is aware of any changes, he notifies the Sender by calling the MSP PUT method. The changes might be triggered by the CPO sending additional Charging Profiles, or some local limit being applied to the Charge Point, and the Charge Point notifies the CPO of the Changes.

The Receiver can cancel/remove an existing ChargingProfile, it can let the eMSP know by calling the MSP PUT method.

For calculating optimum ChargingProfiles it might be useful for the eMSP or SCSP to know the ChargingProfile that the Charge Point has planned for the Session: ActiveChargingProfile. The ActiveChargingProfile might differ from ChargingProfile requested via OCPI. There might be other limiting factors being taken into account by the CPO and or Charge Point, that limit the ChargingProfile.

The ActiveChargingProfile profile can be requested by the Sender by calling the CPO GET method on the Charging Profile Receiver interface. The CPO will then ask the Charge Point for the planned ActiveChargingProfile. When that is received it is forwarded to the URL given by the eMSP or SCSP.

The CPO can limit the amount of request that can be done on the Charging Profiles interface, this too prevent creating a too high load or data usages. To do this the CPO can reject a request on the Charging Profile Receiver interface be responding with: TOO_OFTEN.

If the Sender (typically eMSP or SCSP) wants to have a reference between the calls send to the Receivers interface and the asynchronous result received from the Charge Point via the CPO, the Sender can make some unique identifier part of the`response_url` that is part of every method in the Receiver interface. The Receiver will call this URL when the result is received from the Charge Point. The Sender can then match the unique identifier from the URL called with the request.

3. Calibration support

When you bill per volume, the customer (in this case EV driver) must be able to trust that that volume is right and be able to verify it. For EV charging, this means in most situations where either time (minutes of charging) or kWh or a combination, the registration must be done via a certified, calibrated system and the EV driver must be able to verify that the information on the transaction information after charging is correct.

In Germany, this Calibration law is called “Eichrecht” and the regulators have defined specific requirements. To ensure that EV charging systems are compliant with German Eichrecht, but can also work in other countries, for OCPI so called ‘signed meter values’ are added to the CDRs.

This means that a CDR can (not must) contain a SignedData value via a SingedData Class. This class contains all the information of the signed data. Which encoding method is used, if needed, the public key and a list of signed values.

For the German Eichrecht, different solutions are used, all have (somewhat) different encodings. Below the table with known implementations and the contact information for more information.

Name	**Description**	**Contact**
OCMF	Proposed by SAFE	https://has-to-be.com
Alfen Eichrecht	Alfen Eichrecht encoding / implementation	https://alfen.com/de/downloads
EDL40 E-Mobility Extension	eBee smart technologies implementation	https://www.ebee.berlin
EDL40 Mennekes	Mennekes implementation	

Part of the SignedData is the SignedValue Class. This class contains the signed and the plain/unsigned data. By decoding the data, the receiver can check if the content has not been altered.

3. OCPI 3.0

To make OCPI easier to use and that way offer better and more efficient services, in the ECISS project the total structure and setup of OCPI (architecture, specifications and documentation) has been examined. Currently the structure described here is proposed to the community.

3.1 OCPI 3.0 setup/outline

The following new overall structure is proposed

- Foreword and explanation of structure (index)
- Introduction to roaming, OCPI and use of it in extended environment (i.e. explaining the landscape)
- Use Cases
- Transport Modules (specs)
- Swagger files for supporting implementations
- Test cases description

3.2 Main improvements OCPI 3.0

Besides a better setup of how OCPI is 'offered', like explained above there will be the following main changes/improvements compared to current version 2.2.

1. Additional / improved modules
   E.g: V2G support, 15118 support, improved other desired and need modules
2. More efficient checking for valid tokens
   Currently tokens can either be checked via a whitelist or via a real time check to all connected MSPs
   This process will be improved
3. Less dependency between modules
   a. Modules have references to each other data but should not be via enforced links be dependent on each other

3.3 Status

OCPI v3.0 will be a major change and upgrade. This requires not only huge effort to design it, but also for companies to implement it. Most companies at this moment have implemented OCPI v2.1.1 and are busy with implementation of v2.2. As described in this chapter for OCPI v3.0 in the ECISS project first major steps have been set. It is now to other projects and to the EVRoaming Foundation to take it to the next level and together with the OCPI community and contributors get into the next phase of discussing and designing OCPI v3.0.