# How to make electric vehicle chargers more reliable

## Comparing IEC 63110-2 design concepts to OCPP version 2.0.1 and 2.1

Brussels, 16 June 2025


Authors

- Luka De Bruyckere, Senior Programme Manager – ECOS luka.debruyckere@ecostandard.org
- Tomi Engel, Technical Expert – ECOS tomi@objectfarm.org



Acknowledgements

- Alison Grace, Senior Press & Communication Manager – ECOS


## Contents

## Executive summary

The rapid growth of electric vehicles (EVs) means that we need far more public and private charging stations. Charging station operators often manage large networks, relying on communication systems for handling access to each – as well as payment of charging sessions. Different types of charging stations and their management systems must be compatible to appropriately facilitate the transition to EVs. This paper identifies which methods can achieve this in the best way.

There are various tools that could be used to ensure compatibility between different types of charging stations and charging station management systems. For example, the Open Charge Alliance (OCA) developed its **Open Charge Point Protocol (OCPP)**. At the same time, a working group of the International Electrotechnical Commission (IEC) started drafting the **IEC 63110** series, a global communication standard between charging stations and Charging Station Operators.

This paper compares IEC 63110 with the latest versions of OCPP (version 2.0.1 and version 2.1), showing that OCPP is missing some important features. Going forward, this must be rectified to ensure that charging station networks are as effective as possible. We map out a path to make this happen.

### Differences between OCPP and IEC 63110

IEC 63110 had the advantage of starting from a clean slate, as opposed to OCPP, which evolved over multiple releases that needed to remain compatible with earlier versions. While new features can be added to OCPP in a backward-compatible way, this sometimes increases complexity, leading to higher development, operational, and computational costs.

Charging station networks are essentially large-scale distributed computer systems. Temporary communication outages — for instance, due to component failures — will inevitably affect these networks. Given the growing importance of well-functioning charging stations for society, they should be designed in ways to limit the impact of unavoidable failures. IEC 63110 is based on such a "design for failure" approach.

In contrast to OCPP, which only defines the communication interface of a charging station, IEC 63110 defines a communication architecture with multiple actors, including for energy management. The goal of IEC 63110 was to make charging station management system deployments more robust, more reliable, and easier to scale compared to OCPP. We recommend OCPP to take over specific conceptual design choices from IEC 63110. This way communication can

- be more reliable in case of connection failures, with a reduced implementation effort;
- ensure more targeted information access, helping to avoid communication overload;
- allow the system to recover from software crashes more easily;
- allow for a smoother and simpler integration with energy management systems.

### Applying lessons learned

IEC 63110 applied the lessons learned from real-world OCPP deployments and well-established electrical power network standards such as IEC 61850[i]. However, IEC unfortunately decided to halt the development of IEC 63110 once OCPP 2.0.1 became an official IEC standard.
Given these developments, the lessons learned while designing IEC 63110 mustn't be lost. An OCPP 3.0 series could take advantage of this – we outline exactly how in this paper and recommend that OCPP authors take all relevant learnings into account when further developing and improving OCPP.

[i] The IEC 61850 series defines communication protocols to provide communication between equipment located in an electric substation.

# Introduction

The breakthrough of electric vehicles over the last decade requires a rapid increase of both public and private charging stations (CS).

Charging stations operators (CSOs) run large networks of up to millions of public charging stations per operator. To control the access to public charging stations and to enable proper payment of the provided services, CSOs depend on management systems based on machine-to-machine communication. From an information technology perspective a CSO is operating a large scale distributed computer system.

Each CSO has a custom charging station management system (CSMS) that needs to connect to different types of charging stations. This led to the necessity to develop a common protocol for the communication between the charging stations and the CSMS. The development of the Open Charge Point Protocol (OCPP) by the Open Charge Alliance (OCA) was started in 2009. OCPP was a pragmatic solution which rapidly evolved over many versions. Instead of using technical strategies from industrial engineering, it was based on technologies that are common in online services for the World Wide Web. This simplified the implementation because it could tap into a large pool of related software tools and developers which are familiar with those tools and technologies (e.g. HTTPS, JSON, SOAP, WebSockets).

Meanwhile, the International Electrotechnical Commission (IEC) decided many years ago that an international standard for a communication protocol for the information exchange between charging infrastructures and charging stations operators is needed. At first an extension of the widely used IEC 61850 protocol series was proposed, which was already used for the industrial grade control of substations and other components of today's electrical power networks. However, due to the technical complexity of the IEC 61850 protocol this approach did not get any recognition and IEC 61850-90-8 did not see any adoption.

Based on that realisation the committees within IEC decided to take a new approach, by forming a new working group (JWG11) including experts from the electric vehicle side (TC69) in addition to the electrical power network experts (TC57). JWG11 was given the task to combine the lessons learned from industrial protocols like IEC 61850 with the lessons learned from the real-world deployment of OCPP based systems. This effort resulted in the IEC 63110 document series. JWG11 included experts working on or with OCPP, and developed technical solutions to make CSMS deployments more robust, more reliable and easier to scale.

In October 2024 IEC TC 69 decided to halt the development of IEC 63110. OCPP 2.0.1 will become an IEC standard. Given these developments it is key to preserve the valuable features of IEC 63110. This paper compares the design concepts of IEC 63110 with the latest versions of OCPP (version 2.0.1 and version 2.1), listing the most important features currently missing from OCPP.

# Document structure

The first chapter provides an overview of the key design approaches of the IEC 63110 protocol.

The second chapter outlines some key differences between the latest versions of OCPP and the design approaches of IEC 63110.

Annex A provides a brief introduction to some fundamental aspects of distributed computer systems. This is intended to help readers without a computer science background to better understand some of the technical aspects and design decisions of the different protocols.

## Abbreviated terms

The following technical terms are used in this document in their abbreviated form. For most of their detailed definitions can be found in the documents of the IEC 63110 series.

| Abbreviation | Technical term |
|---|---|
| CEM | customer energy manager |
| CS | charging station |
| CSMS | charging station management system |
| DSO | distribution system operator |
| EC | energy constraints |
| EV | electric vehicle |
| ETP | energy transfer plan |
| HTTPS | hypertext transfer protocol secure |
| IEC | International Electrotechnical Commission |
| IP | internet protocol |
| IPv4 | internet protocol version 4 |
| IPv6 | internet protocol version 6 |
| JSON | JavaScript simple object notation |
| OCA | Open Charge Alliance |
| OCPP | open charge point protocol |
| PC | power constraints |
| PEBC | power envelope based control |
| PRE | power range envelope |
| RM | resource manager |

| | |
|---|---|
| SOAP | simple object access protocol |
| TCP | transmission control protocol |
| XMPP | extensible messaging and presence protocol |

Note: Terms which in this document are typeset in italics usually refer to technical identifiers that are defined in the related specifications or standards.

## The design aspects of the IEC 63110 protocol

IEC 63110-1, the first document published by IEC TC69 JWG11, defined the goals of the CSMS protocol by means of high-level requirements and use cases. A considerable portion of the JWG11 experts were OCA members, and some actively participated in drafting the OCPP standard. This allowed JWG11 to identify real-world issues of the existing solutions while learning about the reasons for design decisions in OCPP, avoiding certain design pitfalls.

Based on its high-level goals, JWG11 started to develop the concrete CSMS protocol architecture within the IEC 63110-2 document. One of these goals was that IEC 63110 should be a "superset" of the existing OCPP protocol, meaning that a mapping from OCPP to IEC 63110 should be possible without loss of significant information. Actual backward compatibility was not possible, due to copyright reasons, nor desired, given that the JWG11 experts wanted to clean up specific mistakes from OCPP.

In addition, IEC 63110 had to provide a design that would allow the integration (mapping) of object models that were already defined in the field of electrical power grid management (IEC TC57) as well as smart metering (IEC TC13). Qualitative integration with the energy management standards (IEC SC23K) was also required.

Therefore, in contrast to OCPP, which only defines the communication interface of a charging station, IEC 63110 defined a communication architecture with multiple actors. Besides defining the application layer messages and the underlying object model of a charging station, IEC 63110 also defined the necessary aspects of a resource manager (RM) for energy management, as well as a communication layer that was not based on a point-to-point paradigm but rather a message broker architecture.

### Overview of essential aspects

Below we list a selection of concepts that were defined within the concrete protocol specification in the IEC 63110-2 draft document. Some of these concepts are part of OCPP as well, either similar or in different forms.

Readers who are not familiar with the technical aspects of distributed computing and communication protocols may find it hard to understand the relevance of the following enumeration. For that reason, some of the more important points will later be picked up again and explained in more detail in dedicated sections. The list below mainly serves as an overview for technical experts who are familiar with this field:

1. Separation of hardware control from business logic

The message design tries to move all customer specific business logic (decision making) into the CSMS. This enables CS products which require less customer specific modifications (e.g. via concepts like the *OperationalPermissionKind*)

2. Message broker architecture

A broker concept at the communication layer allows to support **hierarchical deployment strategies** (e.g. local and cloud CSMS). The XMPP standard was selected for IEC 63110. It can **help in traffic shaping by allowing to reshuffle the order of messages** to meet individual timing requirements. It also helps to bundle traffic into a limited number of TCP ports, which allows **better scaling and simplifies firewall strategies**.

3. Unified object model

It can express the complete device model (state) because it can carry any kind of complex data type and because all data that gets exchanged in messages is clearly mapped into the device state. This is the basis for generic access messages (read state, write state) which allows **full state resynchronization between CSMS and CS**.

4. Composable device model (*Components*)

The unified object model supports arbitrary device models for each individual product, while still supporting predictable, automated validation based on schema matching and type inference.

5. Clearly defined concept for non-standard extensions

Concrete requirements prevent namespace collisions between different vendors when they need to introduce **non-standard extensions** of the object model. This also allows to automatically import (map) object models from other standards (e.g. IEC TC13 "OBIS", IEC TC57 "IEC 61850-7-420").

6. Generic key-path based identification of properties in the unified object model

There is one concept that can be applied in multiple contexts to support precise identification of information (values).

7. Full observability via generic monitoring of the object model

The unified object model combined with the generic key path allows to support very **detailed remote observability** of all aspects of the state of a CS.
This can help in many different contexts: e.g. for **debugging** system issues or to support **traffic shaping strategies**.

8. Detailed error reporting at the message level

Very **detailed error messages** can be provided to support **debugging** procedures and better automation of error classification and recovery.

9. Clearly defined information transaction[ii] concepts

The protocol has support for atomic, ordered state changes for more predictable results. Batched messages allow to compose multiple focused messages into one transaction, which can guarantee the

ii IEC 63110 and this document use the term transaction as the generic concept in computing for atomic and well-defined state changes. OCPP, however, uses the term to describe aspects related to charging sessions and the (potential) billing procedures.

order of changes and **reduce communication latency**. This is especially necessary in a message broker architecture due to potential out-of-sequence message deliveries. However, even within a single message this helps to ensure the **correctness of multi-value updates**.

10. Support for small embedded systems

Type definitions are based on "native" (CPU aware) **simple data types**, which allows efficient storage. Chunked messages and multipart messages allow to **reduce the message buffer size** (e.g. to 10 to 64 kB).

11. Generic "Task" abstraction

There is one simple concept to remote-control long running "tasks" inside a CS. This **simplifies the observability of a CS from the CSMS.**

12. Generic file (BLOB) caching concept

This allows to **reduce the necessary bandwidth** of data transfers as can prevent unnecessary (multiple) transfers of the same data. Multiple use cases (e.g. firmware updates, display configuration) are covered with one set of messages.

13. Generic "power envelope based control" of energy transfer procedures (use cases)

A common representation, based on the "power envelope based control" of EN 50491-12 and IEC 63402, **simplifies the implementation of an energy management strategy** within the CSMS. The same data types (ETP, PRE, PC, EC) can be used to control different charging protocols.

14. Clearly defined integration with premises energy management

The energy management concepts of EN 50491-12 and IEC 63402 are reflected by defining the messages related to the resource manager role (interface) of the premises.

15. End-to-end communication security

Requirements have been defined for first class support of end-to-end information trust. For example generic payload signatures and payload encryption have been considered, and detection of replay attacks could be implemented by utilizing the unique end-to-end identifiers of the message headers.

16. 16. Role based access control security

Requirements have been defined for the XMPP message broker as well as the individual actors to implement role based access control (RBAC). A generic *PropertyAccessKind* concept was defined to support the communication of RBAC related information restrictions. Access control during the *JoinNetwork* phase was also supported by requirements for state of the art credentials and encryption techniques.

## Message broker communication layer

IEC 63110 does not just define the interface of a CS but rather defines the interfaces within the communication network consisting of a CS, a RM and a CSMS. The CSMS itself can be based on a hierarchical deployment of multiple CSMS software instances, each with a different responsibility. However, IEC 63110 does not prescribe the design of the CSMS. It mainly provides the communication protocol to build different interoperable products.

To enable plug-and-play interoperability of systems that are based on IEC 63110, the communication layer must be based on a **message broker architecture**. For this the XMPP standard was selected after

an extensive evaluation process and a vote amongst the experts. One argument for selecting XMPP was that it is already being used in the context of IEC 61850 based electric power grid management. Furthermore, several different implementations of XMPP exist, with different guarantees for message delivery robustness, different failover strategies, and different options for integration with other messaging protocols. This allows developers to pick the implementation that fits best for a given purpose, without the need to invest into a customised implementation.

From a conceptual point of view a message broker design offers several advantages to simple point-to-point protocols. Firstly, messages within a broker-based architecture carry additional information, allowing the message broker to use alternative delivery routes to the final destination (the recipient address). This increases reliability, for instance when a connection would fail. Assigning priority to the information allows to reshuffle the order of message deliveries if e.g. communication links are too saturated. Special broadcast recipient addresses allow to logically send one message to multiple recipients without having to transmit the data multiple times, which can reduce the saturation of a communication link. Another benefit is that in a message broker environment the number of necessary (exposed) TCP ports can also be reduced, which can help to simplify firewall strategies at remote sites.

Overall, a robust and well tested message broker software can reduce the implementation effort of a robust CSMS communication solution, just like a good database reduces the effort for reliable and fast information storage.

## Unified object model

The IEC 63110-2 draft document section 9 describes its approach to defining the information (state) exchange at the application layer as follows:

*This standard is following the approach of a unified application layer object model. This combines the complex value types which define the state represented by a specific device object model (typically for a charging station) with the parameters that are communicated by the message object model of the application layer. This means that many message parameters will be reflected at some location of the device object model as state information.*

While many low-level device control protocols only define state based on simple values (a number or a character sequence) the state of a device in IEC 63110 is described by complex types, which group multiple simple values together according to a schema (a pattern description). This allows to define the representation of complex information like e.g. an energy transfer plan. IEC 63110 thereby adopts the same approach as IEC 61850 and other high-level protocols.

Messages in IEC 63110 are mainly a concept to ensure transactional integrity, which means that for certain steps in a use case, certain information needs to be provided and processed as a group. If a message is not received completely, it cannot be processed, because the data is identified as incomplete. In that case the CS produces an error message. After a message has been processed, the essential parameters will become visible within the device object model at well-defined positions. Each position is a logical address which can be defined with generic key-path based identification.

The concrete implication of this will become clear in the next two sections.

## Full observability via generic monitoring of the object model

The necessity for full object model observability gets introduced in IEC 63110 as follows:

*"In a distributed system which is used to operate physical devices that may be exposed to harsh environmental conditions it is important for a central management system to be able to observe all relevant remote state. This enables the optimization of deployment strategies and may prevent system failure or premature degradation. But understanding which specific remote state allows what kind of insight is typically hard to predict in advance.*
*[...]*
*A key concern in this context is that the amount of potential object model properties can be very large and related monitoring can result in the need to process extremely large volumes of sampled data. This drives the need to support very focused individual monitoring strategies as well as the need to prevent unnecessary data transfers."*

The unified object model together with key-path based identification (addressing) provides the necessary foundation. IEC 63110 defines *MonitoringTaskSetup* messages that are tightly integrated with a generic information logging and reporting mechanism, based on a generic task administration concept. The logging as well as the reporting is defined in such a way that it allows detailed customisation and prioritisation. This can prevent the local device (usually the CS) or the network communication link from getting overwhelmed.

Defining the generic information processing concepts (monitoring, logging, reporting and task administration) within IEC 63110 not only ensures full observability, but it also makes it easier to move the decision (business logic) about how much data is needed by a CSMS to that CSMS itself, instead of having to adapt a CS product to each new CSMS operator.

## State resynchronisation

Distributed computing systems, such as charging station networks, rely on information exchange (e.g. over wireless signals or cables) between the different parts of the system. External factors, such as power or communication cables that are cut during construction works, affect the reliability of this communication. This can result in temporary communication outages. Furthermore, charging stations are exposed to outdoor climatic events creating significant risks of defects.

Given that large systems such as charging station networks are composed of smaller components that can fail, the probability that (a part of) a system will fail increases along with the size of the system. Charging station networks are very likely to break down at some point for a period of time. Therefore, the parts of a charging station should be designed in ways to protect the larger system. A **"Design for failure" approach** can protect the distributed computer system that operates critical infrastructure like a charging station network.

IEC 63110 is based on such a design for failure paradigm. This is needed because when the CSMS software crashes it may take a while to restart. During that time all the attached CS devices should be able to continue operating. However, once the CSMS is running again it risks having an outdated view on the state of the CS devices, or no information at all. To make proper business decisions the CSMS would need to check the internal state of each CS device. This entails checking large data sets, while the CSMS only needs specific aspects for its decision-making.

Similar to full observability the unified object model of IEC 63110, together with key-path based identification (addressing), provides the necessary foundation.

Based on generic object model interactions (e.g. *PropertyTreeDiscovery*, *PropertyValueUpdate*) the CSMS can compare every state it has cached in its own memory and update it to the latest actual state of each CS device. Since the devices only need to retransmit relevant data, the CSMS can rely on the smallest communication exchange possible to be fully operational again.

Without full state resynchronisation capabilities, the CSMS might report incorrect information upstream, or make mistakes such as reserving a dysfunctional CS for a new customer.

### Generic "power envelope based control" of energy transfer procedures

One of the fundamental responsibilities of a CSMS is the management of all related energy transfers while considering all the external constraints and goals. However, different EVs use different charge operation control protocols (e.g. IEC 61851 for basic charging, ISO 15118-20 for high-level charging). The protocols diverge regarding the level of details they provide, for example the uncertainty of the power constraints. There is much more uncertainty when using IEC 61851 than ISO 15118-20, and charging goals of the charging process. Different charging sites are subject to different curtailment agreements with the premise owner or the DSO controlling the local grid connection point.

Writing a good optimization algorithm for energy management is a hard engineering problem. The task gets even harder if each energy transfer comes with a different representation of the external constraints and goals. For that reason, IEC SC23K defined a small set of generic flexibility control types in the IEC 63402 series. One of them was specifically designed for the purpose of EV charging: power envelope based control (PEBC).

Within IEC 63110 every kind of energy transfer is represented by an energy transfer plan (ETP) and the associated power and energy constraints - power range envelope (PRE), power constraints (PC) and energy constraints (EC). That ETP is based on concepts defined for the "power envelope based control" (PEBC) according to IEC 63402. By using a single representation for all energy transfers the job of implementing the energy management function within the CSMS will become easier. Using the same data model, it might even become possible to reused existing energy management algorithms from other IEC 63402 based solutions.

The PEBC based design also allows for what is known in computer science as **separation of concerns**: the CS will map every different EV charge control protocol to the generic PEBC representations, which removes complexity from the CSMS. The CSMS no longer needs to understand all technical low-level details of EV charging and can easier fulfil its energy management responsibility. It also allows the CSMS to preconfigure or control every CS and every EV charging process with the same generic ETP information.

The CSMS will use PEBC based information to interact with the resource manager of the premises, to support the (often mandatory) integration with the grid connection point, without having to understand the actual protocol which is used for the energy management within the premises.

The concepts of the PEBC are mainly defined by the IEC 63110-2 complex value types: *PowerRangeEnvelopeType*, *PowerConstraintsType*, *EnergyConstraintsType* and *EnergyTransferPlanType*. These concepts are also being defined in IEC 63402-2-1.

## Comparison of OCPP version 2.0.1 and 2.1 to IEC 63110

The work on the Open Charge Point Protocol (OCPP) started in 2009, evolving over a number of iterations. OCPP version 1.5 was based on the SOAP communication protocol, which uses XML based data representation. OCPP 1.6, which is the most widely used release, introduced the option to represent data as JSON and to communicate via WebSockets.

The next major iteration was the OCPP version 2 series, which was first published in 2018. While it preserved many concepts version 2 introduced some changes that broke the backward compatibility

with release 1.6. and required a new major release number. The communication link is now based on WebSockets and JSON data representation.

Version 2.1 was released in early 2025. It remains backward compatible with other version 2 releases. New additions focus on the features necessary for supporting ISO 15118-20 charging.

Below we explain key differences between IEC 63110 and OCPP, mirroring the examples covered in the IEC 63110 section. It is not intended as a complete overview of the OCPP architecture or its features. Unless explicitly mentioned, the descriptions apply to all OCPP version 2 protocols.

## Point-to-point communication layer

By design OCPP only specifies the interface of a CS. In its latest releases it defines WebSockets as the communication layer, which is a point-to-point communication protocol. OCPP is not using or defining a message broker architecture such as XMPP in IEC 63110.

However, OCPP 2.0.1 defines the concept of a "*Local Controller*", which is somewhat similar to the Local-CSMS concept defined by IEC 63110-1. With the *Local Controller* concept OCPP is deviating from the pure definition of a CS interface, and is adding information exchange rules for systems which are upstream of a CS. One of the goals of the *Local Controller* is to allow a local system that is physically close to the charging stations, to take over some responsibilities of a CSMS which should or can only be provided (reliably) on premises. For instance, the Local Controller was added to support reliable DSO initiated power curtailment (see OCPP use case "External Charging Limit with Local Controller").

The *Local Controller* appears to be very similar to a message broker in some of the OCPP diagrams, as it should redistribute information locally. However, it is technically not a message broker. OCPP clearly states that each local CS shall be exposed by an individual WebSocket to the upstream CSMS. This does not allow for reducing the number of TCP connections that get established from a charging site to the upstream network (CSMS), as there is no message multiplexing on the upstream links operated by the *Local Controller*.

While it is technically possible to develop a message broker solution using the WebSocket protocol, the architecture defined by OCPP for the *Local Controller* does not allow for such designs. The reason is not clear from the specification.

Any distributed system needs strategies to handle communication link failures and link saturation conditions. Without a standard message broker communication layer the OCPP CSMS implementations need to come up with their own custom strategies. This can be a strength (e.g. more freedom for each operator) or a weakness (e.g. less interoperability or less choices for well tested solutions). In the past some OCA members already requested the definition of a message broker architecture. However, since OCPP - by design - still tries to limit the specification only to the interface of a CS, elements of an overall communication architecture are not considered part of the protocol's scope.

## Partitioned object models

The most essential representation of state within OCPP is the device object model which is based on components and their variables. Components can be placed in the context of a specific EV supply equipment (EVSE) and one of its charging connectors.

But there is also a second "hidden" device object model, which is exposed in the message model. Some messages allow to access information that is not part of the device (component) object model.

For example, the *FiscalMetering* component offers a cumulative export reading in the recommended *EnergyExportRegister* variable. But there is a reporting mechanism for sampled data which allows reading values that are defined by the *MeasurandEnumType*. The sampled data allows to read the same information under the identifier "*Energy.Active.Export.Register*". But in addition sampled data allows to read "*Energy.Active.Import.CableLoss*" or "*Power.Factor*" or other measured information which is not exposed via variables of the *FiscalMetering* component.

IEC 63110 was designed to avoid some of the issues that arise from how component variables are identified. In OCPP the address of each variable follows a fixed pattern that could be defined as: "*evse-.id.connector-.id.(component)name-.instance.(variable)name-.instance*". This fixed pattern makes processing easy, but on the other hand, it makes the representation of the state of a complex device complicated, because OCPP does not allow the nesting of components within components. For example, OCPP defines a *TemperatureSensor* component, but when another component wants to expose an internal temperature reading, it must apply a trick by placing that value inside a variable called *Temperature*. Countless other examples of such tricks exist, which make the processing of information harder instead of easier.

One could even argue that OCPP has a third object model because the values provided by the component variables can have a complex structure as well. In those cases, string values get exchanged in the messages that are formatted as a list of comma separated values (CSV).

OCA members pointed out to JWG11 that there are still many cases where different CS vendors provide (model) the same device component information in different ways, and there are also cases where there was no agreement on the meaning of certain information. It is important to note that such issues are common to any large specification and should not be considered as resulting from protocol design choices. They are caused by missing requirements and could be fixed in future OCPP protocol versions.

## Observability

In OCPP, the generic monitoring of values exposed via the device component object model is provided via the *SetVariableMonitoringRequest* message. The protocol allows to specify the frequency as well as the conditions that will trigger a monitoring report.

OCPP 2.1 added the *PeriodicEventStream* concept to this mechanism, which may have been inspired by the IEC 63110 monitoring system (e.g. *LogComponent*, *loggerConfig*, *reporterConfig*).

However, the *SetVariableMonitoringRequest* mechanism is limited to the data that is exposed via the device component object model. As explained in the previous section not all information is accessible there.

A “service session” in IEC 63110 terminology, is called a Transaction in OCPP. It captures EV charging service related information (e.g. for billing purposes). OCPP Transaction related data can be observed (monitored, recorded) by using the *TransactionEventRequest*. This includes, but is not limited to, the values measured by the *FiscalMetering* component.

However, certain data (CS device state) cannot be observed in a generic way according to custom needs. Some important data can only be extracted by the CSMS from a CS with dedicated messages (pull access) like *GetTariffsRequest* or *GetChargingProfilesRequest*. This will result in excessive communication if used for observability purposes.

## State resynchronisation

As already pointed out, state resynchronisation and observability depend on a similar foundation. Therefore, similar issues arise. All CS data that is not part of the generic device component object model cannot be read easily by a CSMS using OCPP. This is partly because in OCPP the values of component variables cannot (really) represent complex data structures as they are needed for charging profiles, tariffs or similar information.

OCPP, again, partially depends on the retransmission of certain data from the CS to the CSMS by sending dedicated messages (e.g. *GetInstalledCertificateIdsRequest*). But for some data there seems to be no way to extract all information (e.g. there is no "Get" access to the *NetworkConnectionProfile*, only a *SetNetworkProfileRequest* exists).

OCPP has defined a dedicated *TriggerMessageRequest* where the *MessageTriggerEnumType* defines which messages need to be resent by the CS. This can also be used for the purpose of state resynchronisation.

OCPP 2.1 added a way to define custom triggers so that other important states could also be resynchronised. However, the fundamental problems remain because OCPP does not have a unified object model, but relies on three different concepts, instead of one common method for state resynchronisation.

## Control of energy transfer procedures

OCPP defines messages (e.g. *SetChargingProfileRequest*) that allow to shape the energy transfer to individual charging stations or individual electric vehicles. Together with the Local Controller concept OCPP outlines two different ways for the energy management system of the premises to curtail the charging process (e.g. via *ChargingStationExternalConstraints*). The core data structure for this purpose is the *ChargingProfileType* and *ChargingScheduleType*. By stacking multiple representations of the core data structure, relatively complex power limitations can be sent to the charging station.

The OCPP *ChargingScheduleType* is most closely related to what is called a *PowerRangeEnvelopeType* in IEC 63110. However, the *PowerRangeEnvelopeType* is a focused and generic representation of a power envelope (per phase it sets specific power limits over time). The OCPP *ChargingScheduleType* defines a more complex, less focused concept. It carries additional charge protocol related information which, in OCPP 2.1, will be extended further to even carry tariff data specific for ISO 15118-20 (e.g. *AbsolutePriceScheduleType*, *PriceLevelScheduleType*).

The IEC 63110 abstractions of power constraints (PC) and energy constraints (EC) do not exist in the same generic form in OCPP. Obviously, OCPP defines data that covers related aspects (e.g. via the *ChargingNeedsType*), but it cannot represent the level of flexibility or possibilities, that can be expressed by the generic IEC 63110 structures. This not only affects the management of charging but also the integration with the Customer Energy Manager (CEM) system of the premises.

More importantly, OCPP is not able to express uncertainty of a planned energy transfer, while the ETP of IEC 63110 allows to communicate this via percent probability range (PPR) based predictions. Such information is essential for effective energy management algorithms.

Another major difference is that OCPP uses the concept of charging limits, discharging limits and set-points. In contrast, IEC 63110 adopted the more generic PEBC approach of upper and lower limits. Power envelopes of upper and lower limits allow to describe energy transfer behaviour (e.g. both

limits placed on the same side of the zero-power line), which cannot be expressed by the OCPP structures. However, for some energy market services this will be important.

The control of energy transfer procedures in OCPP is based on very complex data structures which became more complicated over the years and may prove difficult to integrate with an energy management algorithm or to even implement correctly.

## Conclusion

| | IEC 63110-2 | OCPP |
|---|---|---|
| Communication layer | **Message broker architecture**: simplifies firewall strategies and increases reliability when a connection fails | **Point-to-point communication** using a *Local Controller* concept and custom strategies for connection failures providing more freedom for the operator, but also less interoperability |
| Object model | **Unified:** grouping simple values together according to a schema, allowing for the representation of complex information | **Partitioned:** not possible to nest components within components, making the representation of the state of a complex device complicated |
| Observability | **Full observability** via generic monitoring of the object model: allows for detailed customisation and prioritisation. The decision about how much data is needed is left to the CSMS itself, instead of having to adapt a CS product to each new CSMS operator | **Partial observability**: some data cannot be observed in a generic way according to custom needs, which results in excessive communication |
| State resynchronisation capabilities | **"Design for failure" approach** allowing for full resynchronisation based on limited communication exchanges | **Retransmission** of data needed, but not all data can be extracted again, leading to potential mistakes |
| Energy transfer procedures | **"Power envelope based control"**: generic, abstract flexibility control type for easier implementation of the energy management function | **No abstraction** and therefore more complex data structures for energy management |

Table 1 – summary of different concepts in IEC 63110-2 and OCPP

Starting with a clean sheet allowed IEC 63110 to apply the lessons learned from OCPP real life experiences and from well-established electrical power network IEC standards such as IEC 61850. For instance, IEC 63110 was able to adopt a limited set of generic data structures (PC, EC, PRE, ETP) from the domain of energy management standards (IEC 63402) that are very expressive, yet easier to implement, and better suited for energy management algorithms. IEC 63110 was also in a position to adopt a unified object model and generic state monitoring techniques which are fundamental in IEC 61850 solutions.

OCPP, on the other hand, has all characteristics of a protocol that needed to evolve over multiple releases under the burden of backwards compatibility and "time to market" pressure. For example, it lacks a consistent object model or clear state resynchronisation principles. While OCPP started out as a specification for the interface of a charging station — a very narrow scope — the addition of concepts like the Local Controller clearly document that real-world deployments have shown that the efficient and reliable operation of large-scale charging networks requires standards for information exchange which actually are part of the upstream CSMS scope.

We hold the opinion that it is key that the lessons learned during the process of designing IEC 63110 are not lost — especially now that the development of the IEC 63110 standard has been halted. An OCPP 3.0 series could take advantage of those lessons learned. We therefore urge OCPP authors to take these learnings into account when further developing and improving OCPP.

# Annex A - Fundamentals of distributed computer systems

In order to better understand why certain aspects of a communication protocol are important and why specific design features matter, it is necessary to keep some fundamental characteristics of distributed computing and related systems in mind.

## Large systems always fail – communication outages

Every component has a specific mean time to failure. While mechanical components (e.g. fans) will break faster than purely electronic components (e.g. power supplies, data storage chips), electronics will eventually fail too. Given that large systems are composed of smaller components that can fail, the probability that (a part of) a system will fail increases along with the size of the system.

In distributed computing systems information exchange (e.g. over wireless signals or cables) is required by design. External factors, such as power or communication cables that are cut during construction works, affect the reliability of the communication between the parts of a distributed system. This can result in temporary communication outages. The risk of an outage can be minimised by deploying strategies such as partition tolerance (see below).

Public charging stations are exposed to outdoor climatic events. Charging stations can get overheated under the summer sun or suffer from short circuits caused by water entering the CS, with partial or complete failure as a result. Given that many charging sites are in remote areas, remote diagnosis or remote recovery procedures are needed.

In short, since charging stations are very likely to break down at some point, they should be designed in ways to protect the larger system they are part of. A "design for failure" approach is recommended to optimally protect the distributed computer system that operates critical infrastructure like a charging station network.

## Information consistency – data races

Computer systems are based on information, which is usually referred to as data or state. At its core every computer can only copy data from one place to another place (*load and store instructions*) and compare that data to other data to decide which copy or compare operation should be performed next (*conditional jump instructions*). Everything else is the result of extremely fast execution of these copy and compare actions.
The tricky part is the copying of data. Within a computer, each piece of data is copied multiple times before it can be evaluated in the main chip, the CPU. This creates a risk of inconsistency when one program just changed the information on the storage drive while another program is still trying to evaluate the old data. In engineering this is called a race condition, because it is the result of a race against time, where one part of the system is either too slow or too fast. Any attempt to solve this requires coordination (e.g. *locks*) between the concurrent activities of shared data manipulation. In addition, atomic data modification (*transaction*) is needed, that guarantees that either all data is received correctly, or no data is received.

In distributed computer systems this issue is even more pronounced than within a single computer. The task of copying the data from one place to another (e.g. from the CSMS to the CS) requires much more time (minutes or seconds, instead of micro- or nanoseconds) than copying data within one computer, which increases the probability of data race problems. Therefore, communication failure in a distributed system is more likely than within one computer.

The consistency of information is an essential requirement for a distributed computer system that operates critical infrastructure.

## Communication networks

In the context of a charging network the impact of the comparably slow speed of communication networks linking the different parts of a distributed computer system is compounded by the need to install charging stations in remote places. Often only wireless (mobile phone) communication technology can be used to connect the CS systems to the CSMS, usually running in the company's data centres.

Due to the nature of low-level communication techniques (e.g. ethernet physical layer, TCP/IPv6 protocol) intense usage of a communication link (saturation) can result in the loss of data during transmission. Many protocols can solve those technical issues by detection and retransmission of lost data. However, they do this in very different ways. This can result in a significant variability of communication speed and even in the actual loss of data or the failure of the connection between two computers. For example, in rural regions with weak wireless signals, a peak of regular internet usage can have a considerable impact on the saturation of certain mobile communication networks.

Possible solutions for industrial distributed computer systems include techniques for data traffic shaping, where different priorities are assigned to information that needs to be transmitted. Some information is required to reach the receiver (e.g. safety warnings reaching the driver) while other information (e.g. temperature measurements) could be ignored (i.e. dropped) if there is insufficient bandwidth somewhere on the communication path, given that it is difficult to know through which path a data packet will be delivered. A good system needs a high level of observability into the end-to-end communication latency and data loss and retransmission rates.

When depending on unreliable communication networks it is common to move the business logic as close as possible to the point of action. To reduce the risk of unpredictable communication problems

charging stations can deploy a (part of the) CSMS locally, as close as possible to the charging stations at each charging site.

## The CAP theorem

Just like in math or physics, computer science has discovered fundamental laws of information processing. For distributed computer systems one of the most important is the CAP theorem: Consistency, Availability, Partition tolerance.
The theorem claims that any distributed computer system can only be optimised to guarantee two of the three qualities. It is impossible to achieve all three at the same time.

Consistency means that when responding to a request to read data the distributed system will always return the most recent modification of that data.

Availability means that when responding to a request to read data the distributed system will return the correct data or an error.

Partition tolerance means that a distributed system continues to operate even if messages get delayed or dropped by the communication network.

Therefore, it is key to define priorities and decide which trade-offs to make. Without going into the technical details, it is key to note that the implications of the theorem are far reaching and need to be taken into account when designing a protocol or implementing a CSMS, because it operates critical infrastructure.

## Backward compatibility

Every computer system or protocol will evolve over time. This often results in a conflict between two goals: backward compatibility versus a clean new design, which will break backward compatibility.

An initial design is always based on certain assumptions (requirements) that might include a vision of future extensions and the necessary techniques to enable such extensions. In computer technology, everything is the result of a trade-off between competing goals. Therefore, there will always come a point when some desired modifications do not fit the existing or expected patterns.

Usually, this results in what is called "*feature creep*": more and more additions are pushed into or piled on top of a framework that was initially not designed for these new tasks. This is often done through "*hacks*", engineering tricks that allow a system to do new things under the constraints of the original system, by breaking some existing rules or patterns in a creative way. Through backward compatible extensions or hacks most systems can evolve to perform any function. Therefore, the claim that something cannot be done with this system but can be done with the other system is often too simplistic.

However, such hacks come at a cost, such as increased complexity leading to higher development, operational, and computational costs.

When the assumptions on which the initial design was based no longer fit evolving requirements, starting with a clean sheet is common practice. This means breaking backward compatibility, while taking into account the lessons learned from the initial design.